MPC8360E PowerQUICC™ II Pro Integrated Communications Processor Family Reference Manual

MPC8360ERM
Rev. 2
05/2007

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

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

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

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

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

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

Document Number: MPC8360ERM
Rev. 2, 05/2007

Contents

Contents

Contents

Contents

Chapter 5
System Configuration

Contents

Contents

Contents

Contents

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 11
Sequencer

Contents

Contents

Contents

Contents

Chapter 14
Security Engine (SEC) 2.4

Contents

Contents

Contents

Contents

Contents

Chapter 16
DUART

Contents

Contents

Contents

Chapter 20
QUICC Engine Block Control

Contents

Chapter 21
QUICC Engine Multiplexing and Timers

Contents

Chapter 22
Serial Peripheral Interface (SPI)

Contents

Contents

Chapter 25
UCC UART Mode and Asynchronous HDLC

Contents

Chapter 26
UCC BISYNC Mode

Contents

Chapter 27
UCC for Fast Protocols

Contents

Chapter 28
Transparent Controller

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 31
QUICC Engine IEEE1588 Assist

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 33
UTOPIA POS Bus Controller (UPC)

Contents

Contents

Contents

Chapter 35
ATM AAL1 Circuit Emulation Service

Contents

Chapter 36
Serial Interface with Time-Slot Assigner

Contents

Chapter 37
Point-to-Point Protocol (PPP)

Contents

Contents

Contents

Chapter 38
Serial ATM Microcode

Contents

Chapter 39
Enhanced MSP Microcode

Contents

Contents

Contents

Chapter 40
L2 Ethernet Switch

Contents

Contents

Chapter 41
E2AAL2 Microcode

Contents

Contents

Contents

Chapter 43
Inverse Multiplexing for ATM (IMA)

Contents

Contents

Chapter 44
Universal Serial Bus Controller

Contents

Appendix A
MPC8358E

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Table Number	Title	Page Number

About This Book

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

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

Audience

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

Organization

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

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

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

Part II includes the following chapters:

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

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

- Chapter 6, “Arbiter and Bus Monitor,” provides an overview of the arbiter in the MPC8360E device. It also describes configuration, control, and status registers of the arbiter.
- Chapter 7, “e300 Processor Core Overview,” provides an overview of the basic functionality of the e300 processor core and briefly describes how the functional units interact.
- Chapter 8, “Integrated Programmable Interrupt Controller (IPIC),” describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. It also provides a definition of the external interrupt signals and their functions. In addition, the interrupt configuration, control, and status registers are described in this chapter.
- Chapter 9, “DDR Memory Controller,” describes the DDR SDRAM memory controller of the MPC8360E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. Special features like ECC error injection support rapid system debug.
- Chapter 10, “Local Bus Controller,” describes the local bus controller (LBC) of the MPC8360E. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Also, it includes an initialization and applications information section with many specific examples of its use.
- Chapter 11, “Sequencer,” describes how the I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. It also provides address translation on outbound PCI transactions.
- Chapter 12, “DMA/Messaging Unit,” describes the four-channel high speed general-purpose DMA controller of the MPC8360E. The channels share buffer space in the IOS to facilitate the gathering and sending of data. The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This communication unit operates with generic messages and doorbell registers. This block also provides a DMA controller that transfers blocks of data independent of the local processor or PCI hosts.
- Chapter 13, “PCI Bus Interface,” describes the PCI interface, which complies with the *PCI Local Bus Specification*, Rev. 2.3. This chapter provides a basic description of PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification.

- Chapter 14, “Security Engine (SEC) 2.4,” describes the SEC 2.4 that is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the e300 core of the MPC8360E. It is optimized to process all the algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE Std. 802.11i.™
- Chapter 15, “I^2C Interfaces,” describes the inter-IC (IIC or I^2C) bus controller of the MPC8360E. This synchronous, serial, bidirectional, multiple-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8360E powers up in boot sequencer mode which allows the 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 MPC8360E to facilitate boundary-scan testing. The JTAG interface complies with the IEEE Std. 1149.1™ boundary-scan specification.
- Chapter 18, “Delay Lock Loop (DLL),” describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described.

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

- Chapter 19, “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 20, “QUICC Engine Block Control,” 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. The QUICC Engine control registers 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 21, “QUICC Engine 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 (UCC’s, UPC’s, TDM’s, etc.)
- Chapter 22, “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 23, “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 24, “UCC as Slow Communications Controllers,” 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 25, “UCC 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 26, “UCC 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 27, “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, high-speed serials (Ethernet, HDLC, HDLC bus, Transparent), and Ethernet for the first Mile (EFM). Chapter 23, “Unified Communications Controllers (UCCs),” should be read before reading this chapter.
- 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, “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 30, “UCC Ethernet Controller (UEC),” describes the Ethernet IEEE 802.3 protocol, which is a widely-used LAN based on the carrier-sense multiple access/collision detect (CSMA/CD) approach.
- Chapter 31, “QUICC Engine IEEE1588 Assist,” covers the IEEE1588 implementation, which, in the QUICC Engine module, is a hardware-assisted method. The hardware assist includes a time stamp unit to recognize PTP frames and transfer relevant timestamps, and a real time clock with high resolution.
- Chapter 32, “ATM Controller AAL0, AAL1, 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, AAL1 circuit emulation service (CES), AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.

- Chapter 33, “UTOPIA POS Bus Controller (UPC),” covers the UPC (UTOPIA/POS-PHY L2 bus controller), which is the UTOPIA/POS-PHY MAC peripheral of the QUICC Engine block. The QUICC Engine block supports UTOPIA/POS-PHY level 2 for both master and slave modes.
- Chapter 34, “Multi-Channel Controller (MCC),” addresses the multi-channel controller (MCC), which supports up to 256 separate time-division serial channels. The MCC is paired with a serial interface (SI), allowing it to communicate over any of the SI’s 8 time-division multiplexed streams (TDM). Proper programming of the SI and SIRAM is necessary for routing the timeslots within a TDM stream to the appropriate MCC channel at the desired time. Users should be familiar with programming the SI and parallel I/O ports before proceeding.
- Chapter 35, “ATM AAL1 Circuit Emulation Service,” describes implementation of circuit emulation service (CES) using ATM adaptation layer type 1 (AAL1) on the QUICC Engine module and should be used as a supplement to Chapter 32, “ATM Controller AAL0, AAL1, and AAL5.”
- Chapter 36, “Serial Interface with Time-Slot Assigner,” pertains to the serial interface (SI), which manages the routing of eight TDM lines to the QUICC Engine block serial drivers, the MCC and 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 37, “Point-to-Point Protocol (PPP),” describes the QUICC Engine implementation of the point-to-point protocol (PPP), which is compliant with the RFCs 1661, 1662, 1990, 2686, 3153. This chapter covers the functionality and data structures of the PPP protocol, the Multilink-MultiClass PPP (ML-MC) protocol, and the PPP mux protocol.
- Chapter 38, “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 39, “Enhanced MSP Microcode,” describes the QUICC Engine multi service platform derivative (QUICC Engine-EMSP), which adds ATM layer functionality, suitable for the implementation of various network applications, such as a port controller on subscriber line cards, ATM cell processing, cable modem controllers, ATM multiplexing and concentrating systems and other multiservice and multiprotocol applications especially in the branch/enterprise dial access applications.
- Chapter 40, “L2 Ethernet Switch,” describes the L2 Ethernet switch in the QUICC Engine block. This switch offers up to eight connection ports of 10/100 Mbps with MII/RMII Ethernet external ports and one CPU internal port. It also provides VLAN functionality, IGMP snooping, store-and-forward switching operation, and packet-error filtering.
- Chapter 41, “E2AAL2 Microcode,” addresses the implementation of the ATM adaptation layer type 2 (AAL2), compliant with the ITU-T recommendations I.363.2 and I.366.1. This chapter describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR, and is meant to be used as a supplement to Chapter 32, “ATM Controller AAL0, AAL1, and AAL5.”
- Chapter 42, “QMC (QUICC Multi-Channel Controller),” 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 43, “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 44, “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, “MPC8358E,” illustrates the MPC8358E and in particular the differences between it and the MPC8360E as described in this manual.
- Appendix B, “Revision History,” lists the major differences between revisions of this reference manual.
- This 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:

- *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture* (MPCFPE32B)—Describes resources defined by the PowerPC architecture.
- Reference manuals (formerly called user’s manuals)—These books provide details about individual implementations.
- Addenda/errata to reference or user’s manuals—Because some processors have follow-on parts an addendum is provided that describes the additional features and functionality changes. These addenda are intended for use with the corresponding reference or user’s manuals.
- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.

- Technical summaries—Each device has a technical summary that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an implementation's reference or user's manual.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale processors.

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

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
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, $\overline{\textit{core int}}$
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[WPEXT] || TCR[WP]
—	Indicates a reserved bit field in an e300 register. Although these bits can be written to as ones or zeros, they are always read as zeros.

Signal Conventions

$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.
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.

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADB	Allowable disconnect boundary
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BRI	Basic rate interface
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CSB	Coherent system bus
CCSR	Configuration control and status register
CEPT	Conférence des administrations Européenes des postes et télécommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
EEST	Enhanced Ethernet serial transceiver
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
GPR	General-purpose register
GUI	Graphical user interface
HDLC	High-level data link control
I^2C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PCI/X	Abbreviation used to describe operation for both the PCI and PCI-X bus functionality
PCI-X	PCI extended
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
PMA	Physical medium attachment
PMD	Physical medium dependent
POR	Power-on reset
PRI	Primary rate interface
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RIO	Abbreviation occasionally used to refer to the RapidIO interface
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
USB	Universal serial bus
UTP	Unshielded twisted pair

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
VA	Virtual address
ZBT	Zero bus turnaround

Part I
Overview

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

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

Chapter 1
Overview

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

Although this document is written from the perspective of the MPC8360E, most of the material applies to the MPC8358E as well. For information on differences between the MPC8360E and the MPC8358E, see Appendix A, "MPC8358E."

1.1 MPC8360E PowerQUICC II Pro Processor Overview

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

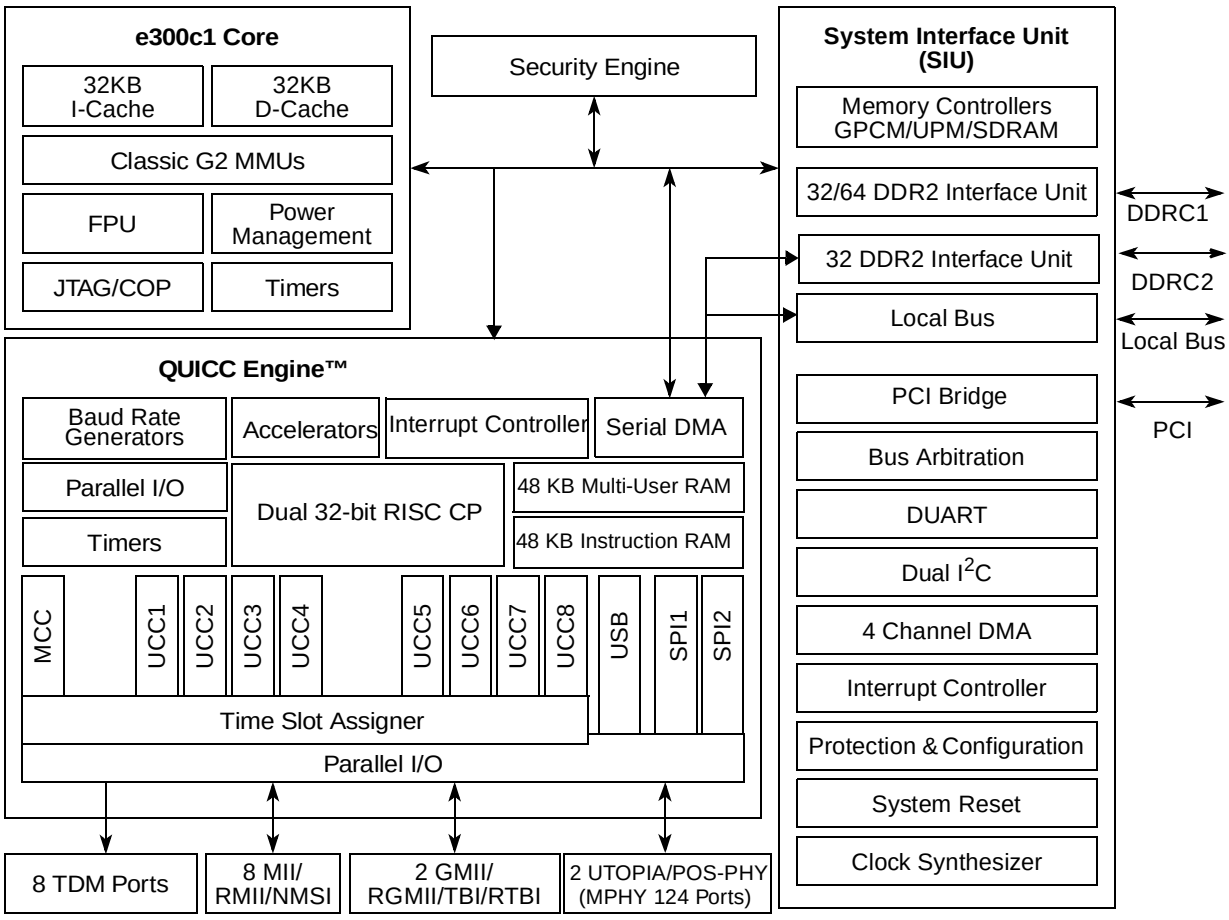


Figure 1-1. MPC8360E Block Diagram

The major features of this device are as follows:

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

- QUICC Engine 2.0 block
  - Two 32-bit RISC controllers for flexible support of the communications peripherals with the following features:
    - One clock per instruction
    - Separate PLL for operating frequency that is independent of system's bus and core frequency for power and performance optimization
    - 32-bit instruction object code
    - Executes code from internal ROM or RAM
    - 32-bit arithmetic logic unit (ALU) data path
    - Modular architecture allowing for easy functional enhancements
    - Slave bus for CPU access of registers and multiuser RAM space
    - 48 Kbytes of instruction RAM
    - 48 Kbytes of multiuser data RAM
    - QE peripheral request interface (for SEC, PCI, and IEEE Std. 1588™)
  - Two serial DMA channels that are optimized for burst transfers and can perform simultaneous accesses to DDR1/DDR2 memory and to the DDR2 memory/local bus
  - Eight universal communication controllers (UCCs) on the MPC8360E and six UCCs on the MPC8358E supporting the following protocols and interfaces:
    - 10/100 Mbps Ethernet/IEEE® Std. 802.3® through MII and RMII interfaces[1]. Note that SMII or SGMII media-independent interface is not supported currently.
    - 1000 Mbps Ethernet/IEEE Std. 802.3 through GMII, RGMII, TBI and RTBI interfaces.
    - 10/100 Mbps Ethernet/IEEE Std. 802.3 L2 switch port through MII and RMII interfaces.
    - IEEE Std. 1588™ support
    - ATM/POS through the UPC
    - Serial ATM through the serial interface
    - HDLC/Transparent (bit rate up to 70 Mbps)
    - HDLC BUS (bit rate up to 10 Mbps)
    - UART
    - BISYNC (bit rate up to 2 Mbps)
    - QUICC multichannel controller (QMC) for 128 TDM channels with 64 channels per UCC
  - PPP, Multi-Link (ML-PPP), Multi-Class (MC-PPP), and PPP multiplexing support
  - One multichannel communication controller (MCC) supporting the following (MPC8360E only):
    - 256 HDLC or transparent channels
    - 128 SS#7 channels
    - Transparent, HDLC or SS#7 per channel
    - Channel multiplexing on up to eight TDM interfaces

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

- ATM controller
  - Full duplex SAR protocols at OC-12 rate through UTOPIA L2
  - AAL5, AAL2, AAL1, AAL0 protocols. TM 4.1 CBR, VBR, UBR, UBR+ traffic types
  - 64 K external connections
  - Inverse multiplexing ATM capability (IMA)
  - 2 UTOPIA/POS-PHY L2 bus controllers (UPC) for MPC8360E and one UPC for MPC8358E supporting 124 ports (optional 128 ports with extended address)
- Universal serial bus (USB) controller
  - USB 1.1 full/low rate compatible
  - USB 2.0 full/low rate compatible (not high speed)
  - USB host mode
  - USB slave mode
- Two serial peripheral interfaces (SPIs). SPI2 is dedicated to Ethernet PHY management.
- Time slot assigner and 8 TDM serial interfaces on the MPC8360E and 4 TDM serial interfaces on the MPC8358E
  - Support T1, CEPT, T1/E1, T3/E3, pulse code modulation highway, ISDN primary rate, Freescale inter chip digital link (IDL), and user-defined TDM serial interfaces. Frame synchronization.
  - Independent Rx and Tx routing RAM with 512 routing entries each.
  - Time slot assigner with bit or byte resolution.
  - TDM interfaces have 1-bit mode for E3/T3 rates in clear channel.
- QUICC Engine interrupt controller supports 4 external and 19 internal discrete interrupt sources and 2 interrupt levels in the system IPIC
- Sixteen independent baud rate generators and 30 input pins to clock the UCCs, SI, UPCs, USB, time-stamps and timer.
- Four independent 16-bit timers that can be interconnected as two 32-bit timers
- Interworking functionality:
  - 8-port L2 10/100-Base T Ethernet switch
  - ATM-to-ATM switching (AAL0, 2, 5)
  - TDM-to-ATM, circuit emulation (CES)
  - Additional interworking functions are supplied as RAM-based microcode packages.

- Security engine optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, 802.11i, iSCSI, and IKE processing. The security engine contains four crypto-channels, a controller, and a set of crypto execution units (EUs). The execution units are:
  - Public key execution unit (PKEU) supporting the following:
    - RSA and Diffie-Hellman algorithms
    - Programmable field size up to 2048 bits
    - Elliptic curve cryptography
    - F2m and F(p) modes

  - Programmable field size up to 511 bits
- Data encryption standard execution unit (DEU)
  - DES and 3DES algorithms
  - Two key (K1, K2) or three key (K1, K2, K3) for 3DES
  - ECB and CBC modes for both DES and 3DES
- Advanced encryption standard unit (AESU)
  - Implements the Rijndael symmetric-key cipher
  - Key lengths of 128-, 192-, and 256-bits
  - ECB, CBC, CCM, and counter (CTR) modes
  - XOR parity generation accelerator for RAID applications
- ARC four execution unit (AFEU)
  - Implements a stream cipher compatible with the RC4 algorithm
  - 40- to 128-bit programmable key
- Message digest execution unit (MDEU)
  - SHA with 160-, 224-, or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- Random number generator (RNG)
- Four crypto-channels, each supporting multi-command descriptor chains
  - Static and/or dynamic assignment of crypto-execution units through an integrated controller
  - Buffer size of 256 bytes for each execution unit, with flow control for large data sizes
  -

- Dual DDR SDRAM memory controllers
  - Programmable timing supporting both DDR1 and DDR2 SDRAM
  - Configurable as two 32-bit buses (MPC8360E only) or one 32-/64-bit bus
  - Up to 333-MHz data rate
    - 64-Mbit to 2-Gbit devices with x8/x16/x32 data ports (no direct x4 support)
  - Up to four physical banks (chip selects), each bank up to 1 Gbyte independently addressable
  - Full ECC support (when configured as 2x32-bit DDR memory controllers, both support ECC)
  - On-die termination (ODT) support when using DDR2
  - Driver impedance calibration support (using MDIC signals) when using DDR2
  - Support for up to 16 simultaneous open pages (up to 32 pages for DDR2)
  - Read-modify-write support
  - Sleep-mode support for SDRAM self refresh
  - Supports auto refresh
  - On-the-fly power management using CKE
  - Registered DIMM support

  - 2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL2 compatible I/O for DDR2
- PCI interface
  - PCI specification revision 2.3 compatible
  - 32-bit PCI interface operating at up to 66 MHz
  - PCI 3.3-V compatible
  - Not 5-V compatible
  - Support for host and agent modes. When in host mode, the PCI controllers support external signal isolation, thus enabling power to shut off external devices.
  - Support for PCI-to-memory and memory-to-PCI streaming
  - Memory prefetching of PCI read accesses and support for delayed read transactions
  - Support for posting of processor-to-PCI and PCI-to-memory writes
  - On-chip arbitration, supporting three masters on PCI
  - Selectable hardware-enforced coherency
- Local bus controller (LBC)
  - Multiplexed 32-bit address and data operating at up to 133 MHz
  - Eight chip selects support eight external slaves
  - Up to eight-beat burst transfers
  - 32-, 16-, and 8-bit ports are controlled by an on-chip memory controller
  - Three protocol engines available on a per chip select basis:
    - General-purpose chip select machine (GPCM)
    - Three user programmable machines (UPMs)
    - Dedicated single data rate SDRAM controller
  - Parity support
  - Default boot ROM chip select with configurable bus width (8, 16, or 32 bits)
- Integrated Programmable interrupt controller (IPIC)
  - Functional and programming compatibility with the MPC8260 interrupt controller
  - System interrupt controller supports 8 external, 25 internal discrete interrupt sources and 2 QUICC Engine interrupt levels
  - Support for one external (optional) and seven internal machine check interrupt sources
  - Programmable highest priority request
  - Four groups of interrupts with programmable priority
  - External and internal interrupts directed to communication processor
  - Redirects interrupts to external $\overline{\text{PCI\_INTA}}$ signal when in core disable mode
  - Unique vector number for each interrupt source
- Dual I^2C interfaces
  - Two-wire interface
  - Multiple-master support
  - Master or slave I^2C mode support

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

1.2 MPC8360E Architecture Overview

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

1.2.1 Power Architecture Core

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

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

- The *e300 Power Architecture™ Core Family Reference Manual* (chapters describing the programming model, cache model, memory management model, exception model, and instruction timing) (Document No. E300CORERM)

- The *Programming Environments Manual for 32-Bit Implementations of the PowerPC™ Architecture* (Document No. MPCFPE32B)

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

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

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

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

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

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

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

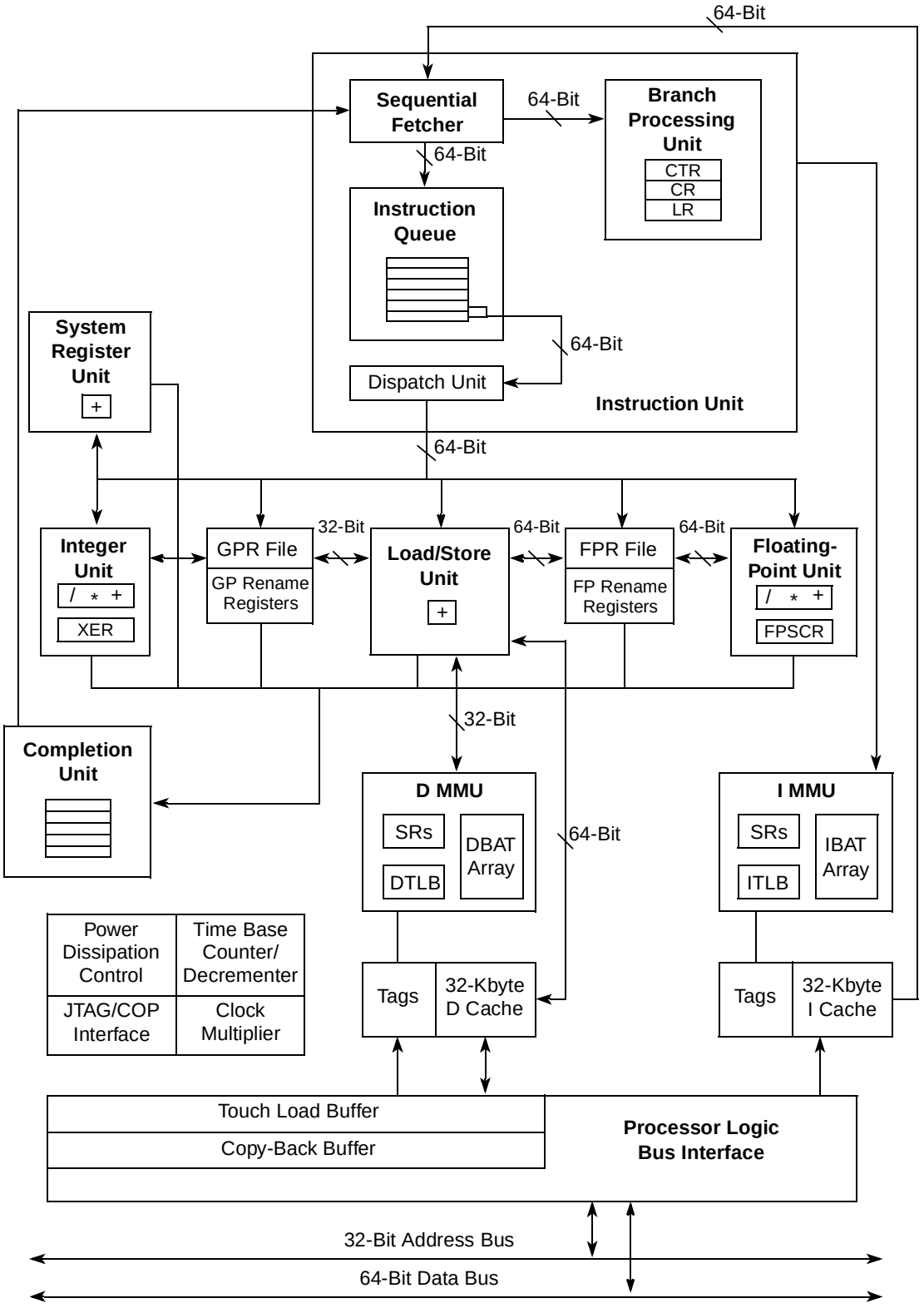


Figure 1-2. MPC8360E Integrated e300c1 Core Block Diagram

1.2.2 QUICC Engine 2.0 Block

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

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

- Eight universal communication controllers (UCCs) on the MPC8360Eand six UCCs on the MPC8358E with the following features:
  - Ethernet, ATM, HDLC/HDLC bus and transparent protocols (also known as fast protocols).
  - UART, BiSync, Async HDLC, Serial ATM and QMC that are user compatible with the SCC of the CPM (also known as slow protocols)
  - Ethernet for the First Mile (IEEE 802.3ah 2BASE-TL and 10PASS-TS)
  - The HDLC and transparent protocols are user compatible with the FCC of the CPM.
- Two UTOPIA-packet over SONET (POS) PHY L2 controllers (UPC) for 124/128 ports (MPC8358E has only one POS PHY L2 controller)
- Two serial peripheral controllers (SPI1 and SPI2[1])
- Multi channel controller (MCC) for 256 channels (MPC8360E only).
- Time slot assigner and serial interface (SI) for 8 TDMs and full duplex routing RAM of 512 entries (MPC8358E has four TDMs).
- One universal serial bus controller (USB 1.1/2.0)

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

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

1. SPI2 can be used only for Ethernet management

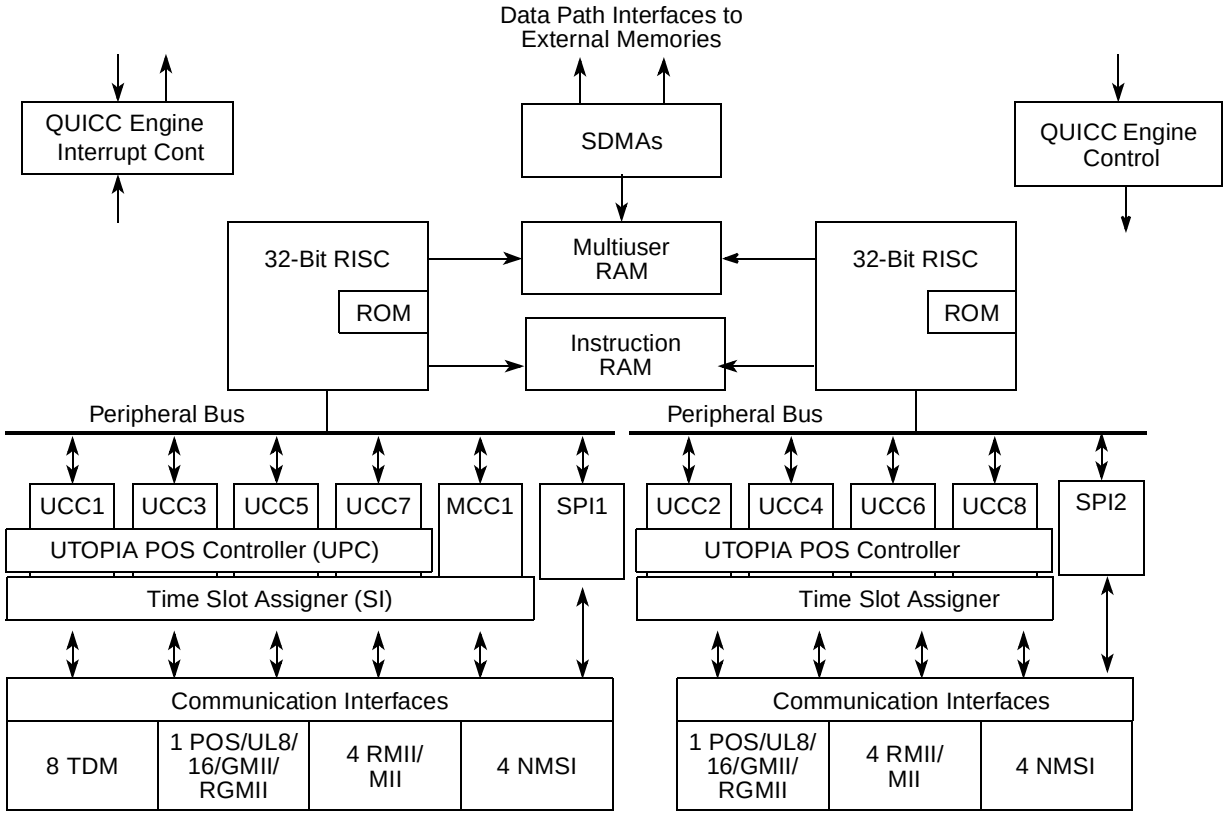


Figure 1-3. QUICC Engine Block Architectural Block Diagram

1.2.2.1 Examples of Chip-Level Pin Multiplexing

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

The first example (DSLAM 48–96 ports[1]) shows that UCC1 and UCC2 are configured for gigabit Ethernet through the RGMII interface. UCC3 and UCC8 are configured for Fast Ethernet through the MII interface. UCC5 and UCC7 are both configured for ATM, provided through the UTOPIA POS controller (UPC1) as a UTOPIA interface with 124 PHYs, and UCC4/UCC6 are likewise configured for ATM through the UTOPIA interface controlled by UPC2. One UART is available through the SIU, SPI1 is used for SPI, and SPI2 is used for Ethernet management; three timers, and two pins remain available for I^2C's. E1/T1 and PCI are not available in this configuration.

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

Table 1-1. Chip-Level Pin Multiplexing Examples

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

The remaining examples in this chapter could be similarly charted.

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

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

- SCC DPLL is not provided
- SCC 10 Base T (7-wire Ethernet) is no longer provided
- HDLC bus protocol programming model is FCC- instead of SCC-compatible
- IDMA I^2C functions are provided as part of the SIU.
- Support for MSP and policer features as standard on the QUICC Engine block

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

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

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

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

- SPI controller implement Ethernet MII serial management interface for up to 32 PHYs.
- Multi-channel controller (MCC) on MPC8360E (MPC8358E does not have an MCC)
  - Multiplexes up to 256 HDLC, transparent, or SS#7[1] channels on one to eight TDM interfaces.
  - Superchannels time-slot of 5,6,7,8, or 16 bits.
  - Optional channel underrun mode: Underrun event disables an individual channel instead of globally disabling the MCC. Per channel soft recovery from underrun is possible.
- Serial interface
  - Support for multi-frame on a single TDM link (TDME)
  - Nibble-parallel data interface on all TDMs.
- TC layer for serial ATM supported as a microcode package.
- IEEE Std. 1588 support
- User can modify the peripheral's (UCC,MCC, SPI) parameter RAM base address in the multi-user RAM
- User programmable FIFO size for UCC fast protocols.
- The QUICC Engine operating frequency is independent of the SIU bus frequency, providing greater performance/power flexibility.
- Two independent time-stamp registers triggered by an external or internal clock

1.2.2.4 Software Migration from the MPC82xx/MPC85xx Family Devices

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

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

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

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

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

1. 128 SS#7 channels

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

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

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

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

1.2.2.5 Serial Protocol Table

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

Table 1-2. QUICC Engine 2.0 Protocols

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

Table 1-2. QUICC Engine 2.0 Protocols (continued)

Protocol	Controller					
	UCC	MCC	SPI	USB	UPC	SI
Ethernet management (SMI)	√ (Opt)	—	√	—	—	—
USB	—	—	—	√	—	—

1.2.2.6 QUICC Engine Configurations

The QUICC Engine block offers configuration flexibility for specific applications. The previously-mentioned functions are all available, but not all of them can be used at the same time. The two physical factors that limit the functionality in any given system are performance and pinout. A pin 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, PKEU, RNG, and RC-4 encryption algorithms in hardware. It also includes XOR parity generation acceleration for RAID applications.

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

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

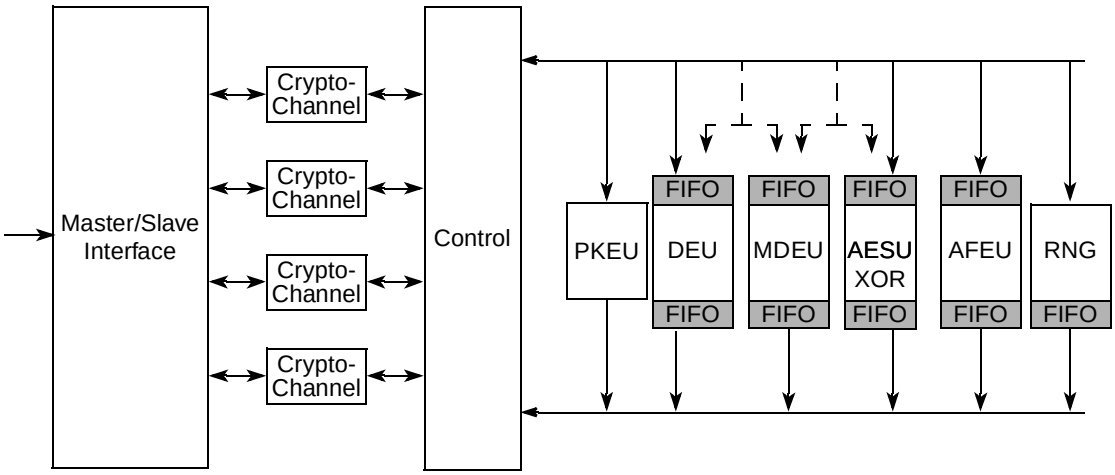


Figure 1-4. Integrated Security Engine Functional Blocks

1.2.4 Dual DDR Memory Controllers

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

The DDR memory controllers include the following features:

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

1.2.5 PCI Controller

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

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

The device PCI controller includes the following distinctive features:

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

1.2.5.1 PCI Bus Arbitration Unit

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

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

1.2.6 Local Bus Controller (LBC)

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

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

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

- Memory controller with eight memory banks (chip selects)
  - 32-bit address decoding with mask
  - Variable memory block sizes (32 Kbytes to 2 Gbytes in FCM mode,)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
  - Parity byte-select
  - Atomic operation
- Synchronous DRAM (SDRAM) machine
  - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
  - Supports up to four concurrent open pages
  - Supports SDRAM port size of 32-, 16-, and 8-bit
  - Supports external address and/or command line buffering
- General-purpose chip-select machine (GPCM)
  - Compatible with SRAM, EPROM, FEPROM, and peripherals
  - Global (boot) chip-select available at system reset
  - Boot chip-select support for 8-, 16-, 32-bit devices
  - Minimum three-clock access to external devices
  - Four byte-write-enable signals ($\overline{\text{LWE}}$[0:3])
- Three user-programmable machines (UPMs)
  - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period
  - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access
  - User-specified control-signal patterns can be initiated by software
  - Support for 8-, 16-, 32-bit devices
  - Page mode support for successive transfers within a burst
- Delay locked loop (DLL) with software-configurable bypass for generating low-frequency bus clocks

1.2.7 Integrated Programmable Interrupt Controller (IPIC)

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

- Functional and programming models are compatible with the MPC8260 interrupt controller
- Support for external and internal discrete interrupt sources, and 2 QUICC Engine interrupt levels
- Support for one external (optional) and seven internal machine checkstop interrupt sources

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

1.2.8 Dual I^{2}C Interfaces

The inter-IC (IIC or I^{2}C) 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^{2}C allows the connection of additional devices to the bus for expansion and system development.

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

The I^{2}C interfaces include the following features:

- Two-wire interface
- Multi-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus
- Address broadcasting supported
- System initialization data is optionally loaded from I^{2}C EPROM by boot sequencer embedded hardware

1.2.9 DMA Controller

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

The DMA controller offers the following features:

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

1.2.10 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The device includes a DUART intended for use in maintenance, bring up, and debug systems. The device provides a standard four-wire handshake (TXD, RXD, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$) 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 equipment including; wireless infrastructure, DSLAMs, routers/switches, line cards, multi-channel modems, network storage, and IADs continue to change and new access technologies are introduced, a programmable communication processing platform that can evolve to accommodate such changes provides equipment vendors with a distinct competitive advantage. These system requirements allow the MPC8360E PowerQUICC II Pro processor to be used in a number of applications as described in the following sections.

1.3.1 SME Router

Figure 1-5 shows a SME router connecting a local area network (LAN) to an ADSL port. The LAN is comprised of an L2 Ethernet switch with VLAN and IGMP snooping support, which is implemented with a microcode package running on four UCCs. See Chapter 40, “L2 Ethernet Switch,” for more details. The ADSL port is implemented with the fifth UCC by enabling the ATM protocols on this peripheral. Together with the Power Architecture core and the system interface unit to a DDR DRAM and a local bus with a memory controller, the MPC8360E with the QUICC Engine block provides a full system solution for this class of product.

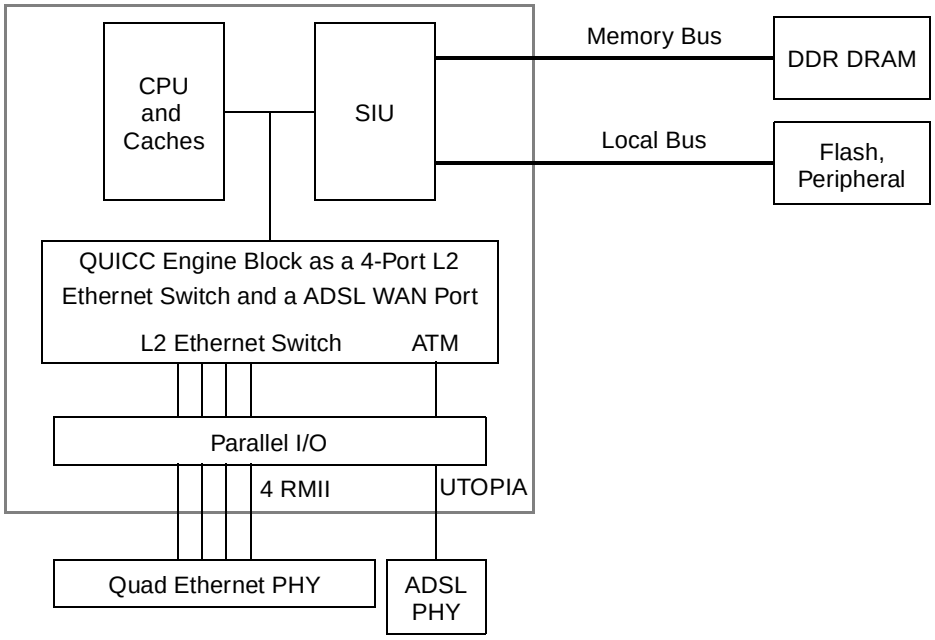


Figure 1-5. QUICC Engine Block in the MPC8360E Used as a SME Router

1.3.2 DSLAM Line Card

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

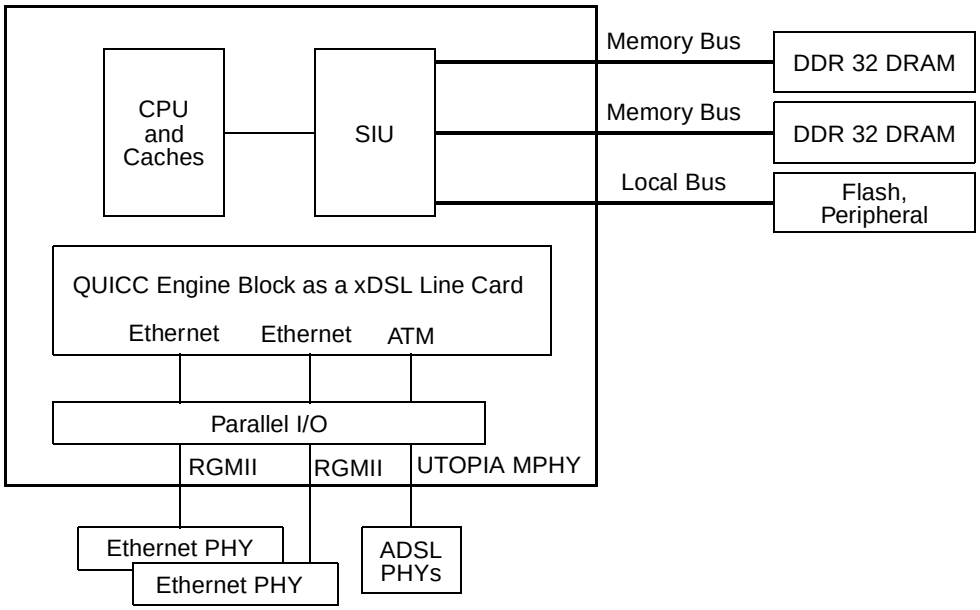


Figure 1-6. QUICC Engine Block in the MPC8360E Used in an DSLAM Line Card

1.3.3 NodeB/BTS—Network Interface Card

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

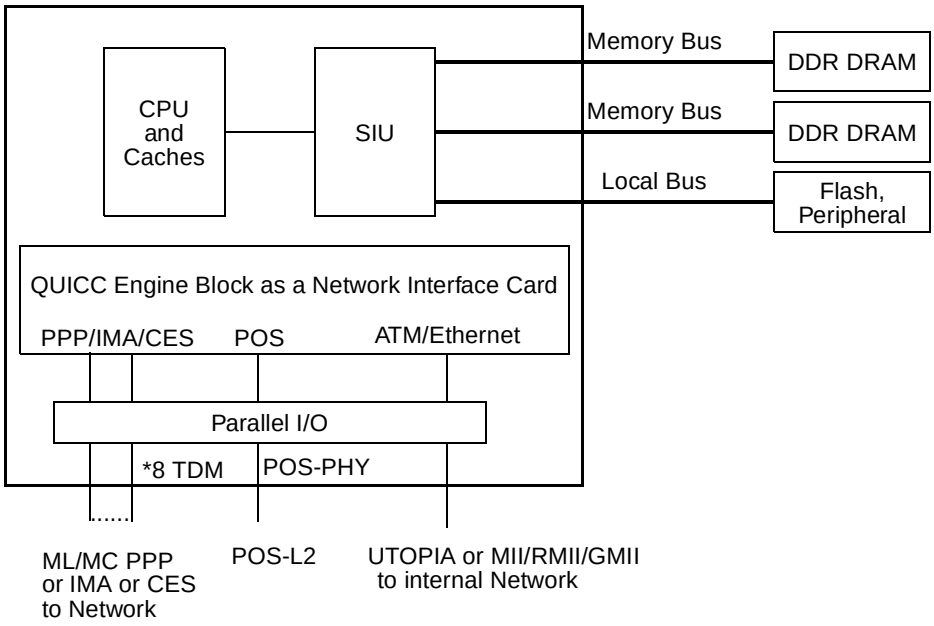


Figure 1-7. Wireless Infrastructure—NodeB/BTS Network Interface Card

Chapter 2
Memory Map

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

2.1 Internal Memory Mapped Registers

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

2.2 Accessing IMMR Memory From the Local Processor

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

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

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

2.3 Complete IMMR Map

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

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

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

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

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

Table 2-1. IMMR Memory Map

Address	Use	Actual Size	Window	Cross Reference
0x00_0000–0x00_01FF	System configuration	512 bytes	512 bytes	Table 2-2
0x00_0200–0x00_02FF	Watchdog timer	16 bytes	256 bytes	Table 2-2
0x00_0300–0x00_03FF	Real time clock	32 bytes	256 bytes	Table 2-2
0x00_0400–0x00_04FF	Periodic interval timer	32 bytes	256 bytes	Table 2-2
0x00_0500–0x00_05FF	Global timers module 1	64 bytes	256 bytes	Table 2-2
0x00_0600–0x00_06FF	Global timers module 2	64 bytes	256 bytes	Table 2-2
0x00_0500–0x00_06FF	Reserved			
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	
0x0_0E00–0x0_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	DLL LBC	20 bytes	256 bytes	Table 2-2
0x00_1200–0x00_12FF	Reserved, should be cleared	—	256 bytes	
0x00_1300–0x00_13FF	Reserved	—	256 bytes	
0x00_1400–0x00_17FF	QUICC Engine parallel I/O ports	168 bytes	1 Kbyte	Table 2-2
0x00_1800–0x00_1BFF	QUICC Engine secondary bus access windows	24 bytes	1 Kbytes	Table 2-2
0x00_1C00–0x00_1FFF	Reserved	—	1 Kbyte	
0x00_2000–0x00_2FFF	DDR MEMC	3.8 Kbytes	4 Kbytes	Table 2-2
0x00_3000–0x00_30FF	I^2C1 controller	24 bytes	256 bytes	Table 2-2
0x00_3100–0x00_31FF	I^2C2 controller	24 bytes	256 bytes	Table 2-2
0x00_3200–0x0_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

Table 2-1. IMMR Memory Map (continued)

Address	Use	Actual Size	Window	Cross Reference
0x00_8300–0x00_837F	PCI configuration	16 bytes	128 bytes	Table 2-2
0x0_8380–0x0_83FF	Reserved	16 bytes	128 bytes	
0x00_8400–0x00_84FF	IOS	256 bytes	256 bytes	Table 2-2
0x00_8500–0x00_85FF	PCI controller	128 bytes	256 bytes	Table 2-2
0x00_8600–0x00_CFFF	Reserved	—	—	
0x00_D000–0x00_DFFF	Secondary DDR MEMC	3.8 Kbytes	4 Kbytes	
0x00_E000–0x02_5FFF	Reserved	—	—	
0x02_6000–0x02_FFFF	Reserved, should be cleared	—	—	
0x03_0000–0x03_FFFF	Security engine	52 Kbytes	64 Kbytes	Table 2-2
0x04_0000–0x0F_FFFF	Reserved, should be cleared	—	—	
0x10_0000–0x1F_FFFF	QUICC Engine 2.0 block	256 Kbytes	1 Mbyte	Table 2-3, Table 2-4

Table 2-2 lists the memory-mapped registers.

Table 2-2. Memory Map

Offset	Register	Access	Reset	Section/Page
System Configuration Registers				
0x0_0000	IMMRBAR—Internal memory map base address register	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved, should be cleared	—	—	—
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	5.2.4.2/5-7
0x0_000C–0x0_001C	Reserved, should be cleared	—	—	—
0x0_0020	LBLAWBAR0—LBC local access window 0 base address register	R/W	0x0000_0000[1]	5.2.4.3/5-8
0x0_0024	LBLAWAR0—LBC local access window 0 attribute register	R/W	0x0000_0000[2]	5.2.4.4/5-9
0x0_0028	LBLAWBAR1—LBC local access window 1 base address register	R/W	0x0000_0000	5.2.4.3/5-8
0x0_002C	LBLAWAR1—LBC local access window 1 attribute register	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0030	LBLAWBAR2—LBC local access window 2 base address register	R/W	0x0000_0000	5.2.4.3/5-8
0x0_0034	LBLAWAR2—LBC local access window 2 attribute register	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0038	LBLAWBAR3—LBC local access window 3 base address register	R/W	0x0000_0000	5.2.4.3/5-8
0x0_003C	LBLAWAR3—LBC local access window 3 attribute register	R/W	0x0000_0000	5.2.4.4/5-9
0x0_0040–0x0_005C	Reserved, should be cleared	—	—	—
0x0_0060	PCILAWBAR0—PCI local access window0 base address register	R/W	0x0000_0000[3]	5.2.4.5/5-10
0x0_0064	PCILAWAR0—PCI local access window0 attribute register	R/W	0x0000_0000[4]	5.2.4.6/5-11
0x0_0068	PCILAWBAR1—PCI local access window1 base address register	R/W	0x0000_0000	5.2.4.5/5-10
0x0_006C	PCILAWAR1—PCI local access window1 attribute register	R/W	0x0000_0000	5.2.4.6/5-11

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0070–0x0_009C	Reserved, should be cleared	—	—	—
0x0_00A0	DDRLAWBAR0—DDR local access window0 base address register	R/W	0x0000_0000[5]	5.2.4.9/5-32
0x0_00A4	DDRLAWAR0—DDR local access window0 attribute register	R/W	0x0000_0000[6]	5.2.4.8/5-13
0x0_00A8	DDRLAWBAR1—DDR local access window1 base address register	R/W	0x0000_0000	5.2.4.9/5-32
0x0_00AC	DDRLAWAR1—DDR local access window1 attribute register	R/W	0x0000_0000	5.2.4.8/5-13
0x0_00B0–0x0_00FC	Reserved, should be cleared	—	—	—
0x0_0100	System general purpose register low (SGPRL)	R/W	0x0000_0000	5.4.3.1/5-25
0x0_0104	System general purpose register high (SGPRH)	R/W	0x0000_0000	5.4.3.2/5-25
0x0_0108	System part and revision ID register (SPRIDR)	R	0x0000_0000	5.4.3.3/5-26
0x0_010C	Reserved, should be cleared	—	—	—
0x0_0110	System priority configuration register (SPCR)	R/W	0x0000_0000	5.4.3.4/5-27
0x0_0114	System I/O configuration register low (SICRL)	R/W	0x0000_0000	5.4.3.5/5-28
0x0_0118	System I/O configuration register high (SICRH)	R/W	0x0000_0000[7]	5.4.3.6/5-30
0x0_011C–0x0_0124	Reserved, should be cleared	—	—	—
0x0_0128	DDR control driver register (DDRCDR)	R/W	0x0004_0000	5.4.3.8/5-33
0x0_012C	DDR debug status register (DDRDSR)	R	0x3300_0000	5.4.3.9/5-35
0x0_0130–0x0_01FC	Reserved	—	—	—
Watchdog Timer (WDT) Registers				
0x0_0200–0x0_0203	Reserved, should be cleared	—	—	—
0x0_0204	SWCRR—System watchdog control register	R/W	0xFFFF_0003 or 0xFFFF_0007[8]	5.5.4.1/5-38
0x0_0208	SWCNR—System watchdog count register	R	0x0000_FFFF	5.5.4.2/5-38
0x0_020C–0x0_020D	Reserved, should be cleared	—	—	—
0x0_020E	SWSRR—System watchdog service register	R/W	0x0000_0000	5.5.4.3/5-39
Real Time Clock Module Registers (RTC)				
0x0_0300	RTCNR—Real time counter control register	R/W	0x0000_0000	5.6.5.1/5-45
0x0_0304	RTLDR—Real time counter load register	R/W	0x0000_0000	5.6.5.2/5-45
0x0_0308	RTPSR—Real time counter prescale register	R/W	0x0000_0000	5.6.5.3/5-46
0x0_030C	RTCTR—Real time counter register	R	0x0000_0000	5.6.5.4/5-46
0x0_0310	RTEVR—Real time counter event register	R/W	0x0000_0000	5.6.5.5/5-47

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0314–0x0_031F	Reserved, should be cleared	—	—	—
Periodic Interval Timer (PIT) Registers				
0x0_0400	PTCNR—Periodic interval timer control register	R/W	0x0000_0000	5.7.5.1/5-52
0x0_0404	PTLDR—Periodic interval timer load register	R/W	0x0000_0000	5.7.5.2/5-52
0x0_0408	PTPSR—Periodic interval timer prescale register	R/W	0x0000_0000	5.7.5.3/5-53
0x0_040C	PTCTR—Periodic interval timer counter register	R	0x0000_0000	5.7.5.4/5-53
0x0_0410	PTEVR—Periodic interval timer event register	R/W	0x0000_0000	5.7.5.5/5-54
0x0_0410–0x0_041F	Reserved, should be cleared	—	—	—
Global Timers Module 1				
0x0_0500	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-61
0x0_0501–0x0_0503	Reserved, should be cleared	—	—	—
0x0_0504	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-61
0x0_0505–0x0_050F	Reserved, should be cleared	—	—	—
0x0_0510	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-64
0x0_0512	GTMDR2—Timer 2 global timers mode register			
0x0_0514	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-66
0x0_0516	GTRFR2—Timer 2 global timers reference register			
0x0_0518	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-66
0x0_051A	GTCPR2—Timer 2 global timers capture register			
0x0_051C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-67
0x0_051E	GTCNR2—Timer 2 global timers counter register			
0x0_0520	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-64
0x0_0522	GTMDR4—Timer 4 global timers mode register			
0x0_0524	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-66
0x0_0526	GTRFR4—Timer 4 global timers reference register			
0x0_0528	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-66
0x0_052A	GTCPR4—Timer 4 global timers capture register			
0x0_052C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-67
0x0_052E	GTCNR4—Timer 4 global timers counter register			
0x0_0530	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-67
0x0_0532	GTEVR2—Timer 2 global timers event register			
0x0_0534	GTEVR3—Timer 3 global timers event register			
0x0_0536	GTEVR4—Timer 4 global timers event register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0538	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-68
0x0_053A	GTPSR2—Timer 2 global timers prescale register			
0x0_053C	GTPSR3—Timer 3 global timers prescale register			
0x0_053E	GTPSR4—Timer 4 global timers prescale register			
Global Timers Module 2				
0x0_0600	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-61
0x0_0601–0x0_0603	Reserved, should be cleared	—	—	—
0x0_0604	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-61
0x0_0605–0x0_060F	Reserved, should be cleared	—	—	—
0x0_0610	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-64
0x0_0612	GTMDR2—Timer 2 global timers mode register			
0x0_0614	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-66
0x0_0616	GTRFR2—Timer 2 global timers reference register			
0x0_0618	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-66
0x0_061A	GTCPR2—Timer 2 global timers capture register			
0x0_061C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-67
0x0_061E	GTCNR2—Timer 2 global timers counter register			
0x0_0620	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-64
0x0_0622	GTMDR4—Timer 4 global timers mode register			
0x0_0624	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-66
0x0_0626	GTRFR4—Timer 4 global timers reference register			
0x0_0628	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-66
0x0_062A	GTCPR4—Timer 4 global timers capture register			
0x0_062C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-67
0x0_062E	GTCNR4—Timer 4 global timers counter register			
0x0_0630	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-67
0x0_0632	GTEVR2—Timer 2 global timers event register			
0x0_0634	GTEVR3—Timer 3 global timers event register			
0x0_0636	GTEVR4—Timer 4 global timers event register			
0x0_0638	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-68
0x0_063A	GTPSR2—Timer 2 global timers prescale register			
0x0_063C	GTPSR3—Timer 3 global timers prescale register			
0x0_063E	GTPSR4—Timer 4 global timers prescale register			
Integrated Programmable Interrupt Controller (IPIC)				

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0700	SICFR—System global interrupt configuration register	R/W	0x0000_0000	8.5.1/8-8
0x0_0704	SIVCR—System global interrupt vector register	R	0x0000_0000	8.5.2/8-9
0x0_0708	SIPNR_H—System internal interrupt pending register	R	0x0000_0000	8.5.3/8-11
0x0_070C	SIPNR_L—System internal interrupt pending register	R	0x0000_0000	8.5.3/8-11
0x0_0710	SIPRR_A—System internal interrupt group A priority register	R/W	0x0530_9770	8.5.4/8-14
0x0_0714	Reserved, should be cleared	—	—	—
0x0_0718	Reserved, should be cleared	—	—	—
0x0_071C	SIPRR_D—System internal interrupt group D priority register	R/W	0x0530_9770	8.5.5/8-14
0x0_0720	SIMSR_H—System internal interrupt mask register	R/W	0x0000_0000	8.5.6/8-15
0x0_0724	SIMSR_L—System internal interrupt mask register	R/W	0x0000_0000	8.5.6/8-15
0x0_0728	SICNR—System internal interrupt control register	R/W	0x0000_0000	8.5.7/8-16
0x0_072C	SEPNR—System external interrupt pending register	R/W	Special	8.5.8/8-18
0x0_0730	SMPRR_A—System mixed interrupt group A priority register	R/W	0x0530_9770	8.5.9/8-18
0x0_0734	SMPRR_B—System mixed interrupt group B priority register	R/W	0x0530_9770	8.5.10/8-19
0x0_0738	SEMSR—System external interrupt mask register	R/W	0x0000_0000	8.5.11/8-20
0x0_073C	SECNR—System external interrupt control register	R/W	0x0000_0000	8.5.12/8-21
0x0_0740	SERSR—System error status register	R/W	0x0000_0000	8.5.13/8-23
0x0_0744	SERMR—System error mask register	R/W	—	8.5.14/8-24
0x0_0748	SERCR—System error control register	R/W	0x0000_0000	8.5.15/8-25
0x0_074C–0x0_074F	Reserved, should be cleared	—	—	—
0x0_0750	SIFCR_H—System internal interrupt force register	R/W	0x0000_0000	8.5.16/8-25
0x0_0754	SIFCR_L—System internal interrupt force register	R/W	0x0000_0000	8.5.16/8-25
0x0_0758	SEFCR—System external interrupt force register	R/W	0x0000_0000	8.5.17/8-26
0x0_075C	SERFR—System error force register	R/W	0x0000_0000	8.5.18/8-27
0x0_0760	SCVCR—System critical interrupt vector register	R	0x0000_0000	8.5.19/8-27
0x0_0764	SMVCR—System management interrupt vector register	R	0x0000_0000	8.5.20/8-28
0x0_0768–0x0_07FF	Reserved, should be cleared	—	—	—
System Arbiter Registers				
0x0_0800	ACR—Arbiter configuration register	R/W	0x0000_0000	6.2.1/6-2
0x0_0804	ATR—Arbiter timers register	R/W	0x00FF_00FF	6.2.2/6-4
0x0_0808	Reserved, should be cleared	R	0x0000_0000	—
0x0_080C	AER—Arbiter event register	R/W	0x0000_0000	6.2.3/6-5
0x0_0810	AIDR—Arbiter interrupt definition register	R/W	0x0000_0000	6.2.4/6-6
0x0_0814	AMR—Arbiter mask register	R/W	0x0000_0000	6.2.5/6-7

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0818	AEATR—Arbiter event attributes register	R	0x0000_0000	6.2.6/6-7
0x0_081C	AEADR—Arbiter event address register	R	0x0000_0000	6.2.7/6-9
0x0_0820	AERR—Arbiter event response register	R/W	0x0000_0000	6.2.8/6-10
0x0_0824–0x0_08FF	Reserved, should be cleared	—	—	—
Reset Module				
0x0_0900	RCWLR—Reset configuration word low register	R	0x0000_0000	4.5.1.1/4-34
0x0_0904	RCWHR—Reset configuration word high register	R	0x0000_0000	4.5.1.2/4-34
0x0_0908–0x0_090C	Reserved, should be cleared	—	—	—
0x0_0910	RSR—Reset status register	R/W	0x0000_0000	4.5.1.3/4-35
0x0_0914	RMR—Reset mode register	R/W	0x0000_0000	4.5.1.4/4-36
0x0_0918	RPR—Reset protection register	R/W	0x0000_0000	4.5.1.5/4-37
0x0_091C	RCR—Reset control register	R/W	0x0000_0000	4.5.1.6/4-38
0x0_0920	RCER—Reset control enable register	R/W	0x0000_0000	4.5.1.7/4-38
0x0_0924–0x0_09FC	Reserved, should be cleared	—	—	—
Clock Module				
0x0_0A00	SPMR—System PLL mode register	R	0x0000_0000	4.5.2.1/4-39
0x0_0A04	OCCR—Output clock control register	R/W	0x0000_0000	4.5.2.2/4-40
0x0_0A08	SCCR—System clock control register	R/W	0xFFFF_FFFF	4.5.2.3/4-41
0x0_0A0C–0x0_0AFC	Reserved, should be cleared	—	—	—
Power Management Control Module				
0x0_0B00	PMCCR—Power management controller configuration register	R/W	0x0000_0000	5.9.3.1/5-73
0x0_0B04	PMCER—Power management controller event register	R/W	0x0000_0000	5.9.3.2/5-74
0x0_0B08	PMCMR—Power management controller mask register	R/W	0x0000_0000	5.9.3.3/5-75
0x0_0B0C–0x0_0BFC	Reserved, should be cleared	—	—	—
QUICC Engine Ports Interrupt Registers				
0x0_0C00–0x0_0C0B	Reserved	—	—	—
0x0_0C0C	CEPIER—QUICC Engine ports interrupt event register	R/W	Undefined	8.5.21/8-29
0x0_0C10	CEPIMR—QUICC Engine ports interrupt mask register	R/W	0x0000_0000	8.5.22/8-30
0x0_0C14	CEPICR—QUICC Engine ports interrupt control register	R/W	0x0000_0000	8.5.23/8-31
0x0_0C18–0x0_0CFF	Reserved	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0D00–0x0_0EFF	Reserved	—	—	—
DLL				
0x0_0F00–0x0_0FFF	Reserved, should be cleared	—	—	—
0x0_1000–0x0_100F	Reserved, should be cleared	—	—	—
0x0_1010	MCK enable register (MCKENR)	R/W	0xFC00_0000	4.5.3/4-42
0x0_1014–0x0_10FF	Reserved, should be cleared	—	—	—
0x0_1100	Reserved. Reset value should be preserved.	R/W	0x0500_0280	—
0x0_1104	Reserved. Reset value should be preserved.	R/W	0x8004_0810	—
0x0_1108	Reserved. Reset value should be preserved.	R/W	0x0000_0000	18.4.1/18-4
0x0_110C	Reserved. Reset value should be preserved.	R	0x0000_0000	18.4.2/18-4
0x0_1110	Reserved. Reset value should be preserved.	R/W	0xFC00_0000	18.4.3/18-5
0x0_1110–0x0_12FF	Reserved, should be cleared	—	—	—
0x0_1300–0x0_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	3.4.2.1/3-21
0x0_1404	CPDATA—QUICC Engine port A data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1408	CPDIR1A—QUICC Engine port A direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_140C	CPDIR2A—QUICC Engine port A direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1410	CPPAR1A—QUICC Engine port A pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1414	CPPAR2A—QUICC Engine port A pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1418	CPODRB—QUICC Engine port B open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_141C	CPDATB—QUICC Engine port B data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1420	CPDIR1B—QUICC Engine port B direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1424	CPDIR2B—QUICC Engine port B direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1428	CPPAR1B—QUICC Engine port B pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_142C	CPPAR2B—QUICC Engine port B pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1430	CPODRC—QUICC Engine port C open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_1434	CPDATC—QUICC Engine port C data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1438	CPDIR1C—QUICC Engine port C direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_143C	CPDIR2C—QUICC Engine port C direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1440	CPPAR1C—QUICC Engine port C pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1444	CPPAR2C—QUICC Engine port C pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_1448	CPODRD—QUICC Engine port D open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_144C	CPDATD—QUICC Engine port D data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1450	CPDIR1D—QUICC Engine port D direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1454	CPDIR2D—QUICC Engine port D direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1458	CPPAR1D—QUICC Engine port D pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_145C	CPPAR2D—QUICC Engine port D pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1460	CPODRE—QUICC Engine port E open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_1464	CPDATE—QUICC Engine port E data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1468	CPDIR1E—QUICC Engine port E direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_146C	CPDIR2E—QUICC Engine port E direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1470	CPPAR1E—QUICC Engine port E pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1474	CPPAR2E—QUICC Engine port E pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1478	CPODRF—QUICC Engine port F open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_147C	CPDATF—QUICC Engine port F data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1480	CPDIR1F—QUICC Engine port F direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1484	CPDIR2F— QUICC Engine port F direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_1488	CPPAR1F—QUICC Engine port F pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_148C	CPPAR2F—QUICC Engine port F pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_1490	CPODRG—QUICC Engine port G open drain register	R/W	0x0000_0000	3.4.2.1/3-21
0x0_1494	CPDATG—QUICC Engine port G data register	R/W	0x0000_0000	3.4.2.2/3-22
0x0_1498	CPDIR1G—QUICC Engine port G direction register 1	R/W	0x0000_0000	3.4.2.3/3-23
0x0_149C	CPDIR2G—QUICC Engine port G direction register 2	R/W	0x0000_0000	3.4.2.3/3-23
0x0_14A0	CPPAR1G—QUICC Engine port G pin assignment register 1	R/W	0x0000_0000	3.4.2.4/3-24
0x0_14A4	CPPAR2G—QUICC Engine port G pin assignment register 2	R/W	0x0000_0000	3.4.2.4/3-24
0x0_14A8–0x0_17FF	Reserved	—	—	—
QUICC Engine Secondary Bus Access Windows Registers				
0x0_1800	LBMCSAR—Local bus memory controller start address register	R/W	0x0000_0000	5.3.2.1/5-18
0x0_1804	SDMCSAR—Secondary DDR memory controller start address register	R/W	0x0000_0000	5.3.2.2/5-19
0x0_1808–0x0_183C	Reserved	—	—	—
0x0_1840	LBMCEAR—Local bus memory controller end address register	R/W	0x0000_0000	5.3.2.3/5-19
0x0_1844	SDMCEAR—Secondary DDR memory controller end address register	R/W	0x0000_0000	5.3.2.4/5-20
0x0_1848–0x0_187C	Reserved	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_1880	LBMCAR—Local bus memory controller attributes register	R/W	0x0000_0000	5.3.2.5/5-20
0x0_1884	SDMCAR—Secondary DDR memory controller attributes register	R/W	0x0000_0000	5.3.2.6/5-21
0x0_1888–0x0_1BFF	Reserved	—	—	—
0x0_1C00–0x0_1FFF	Reserved	—	—	—
DDR Memory Controller Memory Map				
0x0_2000	CS0_BNDS—Chip select memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x0_2008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	
0x0_2010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	
0x0_2018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	
0x0_2020–0x0_207F	Reserved	—	—	—
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x0_2084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	
0x0_2088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	
0x0_208C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	
0x0_2090–0x0_20FF	Reserved	—	—	—
0x0_2100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-14
0x0_2104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-17
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-19
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21
0x0_2114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-23
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-25
0x0_211C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-26
0x0_2120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-26
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-29
0x0_2128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-30
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x0_2134–0x0_2144	Reserved	—	—	—
0x0_2148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-31
0x0_214C–0x0_2BF4	Reserved	—	—	—
0x0_2BF8	DDR_IP_REV1—DDR IP block revision 1	R	0x0002_0200	9.4.1.16/9-31
0x0_2BFC	DDR_IP_REV2—DDR IP block revision 2	R	0x0000_0000	9.4.1.17/9-32

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.18/9-32
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.19/9-33
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.20/9-33
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.21/9-34
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.22/9-34
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.23/9-35
0x0_2E40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.24/9-35
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.25/9-36
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.26/9-37
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.27/9-38
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.28/9-39
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.29/9-39
0x0_2E5C–0x0_2FFF	Reserved	—	—	—
I^2C1 Controller				
0x0_3000	I2C1ADR—I^2C1 address register	R/W	0x00	15.3.1.1/15-5
0x0_3004	I2C1FDR—I^2C1 frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3008	I2C1CR—I^2C1 control register	R/W	0x00	15.3.1.3/15-6
0x0_300C	I2C1SR—I^2C1 status register	R/W	0x81	15.3.1.4/15-8
0x0_3010	I2C1DR—I^2C1 data register	R/W	0x00	15.3.1.5/15-9
0x0_3014	I2C1DFSRR—I^2C1 digital filter sampling rate register	R/W	0x0001_0000	15.3.1.6/15-10
0x0_3018–0x0_30FF	Reserved, should be cleared	—	—	—
I^2C2 Controller				
0x0_3100	I2C2ADR—I^2C2 address register	R/W	0x00	15.3.1.1/15-5
0x0_3104	I2C2FDR—I^2C2 frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3108	I2C2CR—I^2C2 control register	R/W	0x00	15.3.1.3/15-6
0x0_310C	I2C2SR—I^2C2 status register	R/W	0x81	15.3.1.4/15-8
0x0_3110	I2C2DR—I^2C2 data register	R/W	0x00	15.3.1.5/15-9
0x0_3114	I2C2DFSRR—I^2C2 digital filter sampling rate register	R/W	0001_0000	15.3.1.6/15-10
0x0_311C–0x0_31FF	Reserved, should be cleared	—	—	—
0x0_3200–0x0_3FFF	Reserved, should be cleared	—	—	—
DUART				

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Local Bus Controller (LBC) Registers				
0x0_5000	BR0—Base register 0 **Note:** Port size for BR0 is configured from the value of RCWH[ROMLOC] which is loaded during reset, hence 'RR' is either 0x08, 0x10, or 0x18.	R/W	0x0000_RR01	10.3.1.1/10-11
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	10.3.1.2/10-12
0x0_500C	OR1—Options register 1		0x0000_0000	
0x0_5014	OR2—Options register 2			
0x0_501C	OR3—Options register 3			
0x0_5024	OR4—Options register 4			
0x0_502C	OR5—Options register 5			
0x0_5034	OR6—Options register 6			
0x0_503C	OR7—Options register 7			
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	10.3.1.3/10-18
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	10.3.1.5/10-21
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	10.3.1.6/10-22
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	10.3.1.7/10-22
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	10.3.1.8/10-24
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	10.3.1.9/10-25
0x0_50B0	LTESR—Transfer error status register	Read/ bit-reset	0x0000_0000	10.3.1.10/10-25
0x0_50B4	LTEDR—Transfer error check disable register	R/W	0x0000_0000	10.3.1.11/10-27
0x0_50B8	LTEIR—Transfer error interrupt enable register	R/W	0x0000_0000	10.3.1.12/10-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	10.3.1.13/10-29
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	10.3.1.14/10-29
0x0_50D0	LBCR—Local bus configuration register	R/W	0x0000_0000	10.3.1.15/10-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.16/10-31

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_6000–0x0_6FFF	Reserved, should be cleared	—	—	—
DMA Registers				
0x0_8030	OMISR—Outbound message interrupt status register	Special	0x0000_0000	12.4.1/12-4
0x0_8034	OMIMR—Outbound message interrupt mask register	R/W	0x0000_0000	12.4.2/12-6
0x0_8050	IMR0—Inbound message register 0	R/W	0x0000_0000	12.4.3/12-7
0x0_8054	IMR1—Inbound message register 1	R/W	0x0000_0000	12.4.3/12-7
0x0_8058	OMR0—Outbound message register 0	R/W	0x0000_0000	12.4.4/12-7
0x0_805C	OMR1—Outbound message register 1	R/W	0x0000_0000	12.4.4/12-7
0x0_8060	ODR—Outbound doorbell register	R/W	0x0000_0000	12.4.5/12-8
0x0_8068	IDR—Inbound doorbell register	R/W	0x0000_0000	12.4.5/12-8
0x0_8080	IMISR—Inbound message interrupt status register	R/W	0x0000_0000	12.4.6/12-9
0x0_8084	IMIMR—Inbound message interrupt mask register	R/W	0x0000_0000	12.4.7/12-11
0x0_8100	DMAMR0—DMA 0 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x0_8104	DMASR0—DMA 0 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x0_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x0_8110	DMASAR0—DMA 0 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x0_8118	DMADAR0—DMA 0 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x0_8120	DMABCR0—DMA 0 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x0_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x0_8180	DMAMR1—DMA 1 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x0_8184	DMASR1—DMA 1 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x0_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x0_8190	DMASAR1—DMA 1 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x0_8198	DMADAR1—DMA 1 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x0_81A0	DMABCR1—DMA 1 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x0_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x0_8200	DMAMR2—DMA 2 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x0_8204	DMASR2—DMA 2 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x0_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x0_8210	DMASAR2—DMA 2 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x0_8218	DMADAR2—DMA 2 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x0_8220	DMABCR2—DMA 2 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x0_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x0_8280	DMAMR3—DMA 3 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x0_8284	DMASR3—DMA 3 status register	R/W	0x0000_0000	12.4.8.2/12-14

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x0_8290	DMASAR3—DMA 3 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x0_8298	DMADAR3—DMA 3 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x0_82A0	DMABCR3—DMA 3 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x0_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x0_82A8	DMAGSR—DMA general status register	R	0x0000_0000	12.4.8.8/12-18
0x0_82B0–0x0_82FF	Reserved, should be cleared	—	—	—
PCI Software Configuration Registers				
0x0_8300	PCI CONFIG_ADDRESS	W	—	13.3.1.1/13-13
0x0_8304	PCI CONFIG_DATA	R/W	—	13.3.1.2/13-14
0x0_8308	PCI INT_ACK	R	—	13.3.1.3/13-15
0x0_830C–0x0_837F	Reserved, should be cleared	—	—	—
0x0_8380–0x0_83FF	Reserved	—	—	—
Sequencer (IOS)				
0x0_8400	POTAR0—PCI outbound translation address register 0	R/W	0x0000_0000	11.4.1/11-3
0x0_8408	POBAR0—PCI outbound base address register 0	R/W	0x0000_0000	11.4.2/11-3
0x0_8410	POCMR0—PCI outbound comparison mask register 0	R/W	0x0000_0000	11.4.3/11-4
0x0_8418	POTAR1—PCI outbound translation address register 1	R/W	0x0000_0000	11.4.1/11-3
0x0_8420	POBAR1—PCI outbound base address register 1	R/W	0x0000_0000	11.4.2/11-3
0x0_8428	POCMR1—PCI outbound comparison mask register 1	R/W	0x0000_0000	11.4.3/11-4
0x0_8430	POTAR2—PCI outbound translation address register 2	R/W	0x0000_0000	11.4.1/11-3
0x0_8438	POBAR2—PCI outbound base address register 2	R/W	0x0000_0000	11.4.2/11-3
0x0_8440	POCMR2—PCI outbound comparison mask register 2	R/W	0x0000_0000	11.4.3/11-4
0x0_8448	POTAR3—PCI outbound translation address register 3	R/W	0x0000_0000	11.4.1/11-3
0x0_8450	POBAR3—PCI outbound base address register 3	R/W	0x0000_0000	11.4.2/11-3
0x0_8458	POCMR3—PCI outbound comparison mask register 3	R/W	0x0000_0000	11.4.3/11-4
0x0_8460	POTAR4—PCI outbound translation address register 4	R/W	0x0000_0000	11.4.1/11-3
0x0_8468	POBAR4—PCI outbound base address register 4	R/W	0x0000_0000	11.4.2/11-3
0x0_8470	POCMR4—PCI outbound comparison mask register 4	R/W	0x0000_0000	11.4.3/11-4
0x0_8478	POTAR5—PCI outbound translation address register 5	R/W	0x0000_0000	11.4.1/11-3
0x0_8480	POBAR5—PCI outbound base address register 5	R/W	0x0000_0000	11.4.2/11-3
0x0_8488	POCMR5—PCI outbound comparison mask register 5	R/W	0x0000_0000	11.4.3/11-4
0x0_84F0	PMCR—Power management control register	R/W	0x0000_0000	11.4.4/11-5
0x0_84F8	DTCR—Discard timer control register	R/W	0x0000_0000	11.4.5/11-6

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
PCI Error Management Registers				
0x0_8500	PCI_ESR—PCI error status register	R / w1c	0x0000_0000	13.3.2.1/13-15
0x0_8504	PCI_ECDR—PCI error capture disable register	R/W	0x0000_0000	13.3.2.2/13-16
0x0_8508	PCI_EER—PCI error enable register	R/W	0x0000_0000	13.3.2.3/13-17
0x0_850C	PCI_EATCR—PCI error attributes capture register	R/W	0x0000_0000	13.3.2.4/13-18
0x0_8510	PCI_EACR—PCI error address capture register	R	0x0000_0000	13.3.2.5/13-19
0x0_8514	PCI_EEACR—PCI error extended address capture register	R	0x0000_0000	13.3.2.6/13-20
0x0_8518	PCI_EDCR—PCI error data capture register	R	0x0000_0000	13.3.2.7/13-20
0x0_851C	Reserved	—	—	—
PCI Control and Status Registers				
0x0_8520	PCI_GCR—PCI general control register	R/W	0x0000_0000	13.3.2.8/13-21
0x0_8524	PCI_ECR—PCI error control register	R/W	0x0000_0000	13.3.2.9/13-21
0x0_8528	PCI_GSR—PCI general status register	R	0x0000_0000	13.3.2.10/13-22
PCI Inbound ATU Registers				
0x0_8538	PITAR2—PCI inbound translation address register 2	R/W	0x0000_0000	13.3.2.11/13-23
0x0_853C	Reserved, should be cleared	—	—	—
0x0_8540	PIBAR2—PCI inbound base address register 2	R/W	0x0000_0000	13.3.2.12/13-23
0x0_8544	PIEBAR2—PCI inbound extended base address register 2	R/W	0x0000_0000	13.3.2.13/13-24
0x0_8548	PIWAR2—PCI inbound window attributes register 2	R/W	0x0000_0000	13.3.2.14/13-24
0x0_8550	PITAR1—PCI inbound translation address register 1	R/W	0x0000_0000	13.3.2.11/13-23
0x0_8554	Reserved, should be cleared	—	—	—
0x0_8558	PIBAR1—PCI inbound base address register 1	R/W	0x0000_0000	13.3.2.12/13-23
0x0_855C	PIEBAR1—PCI inbound extended base address register 1	R/W	0x0000_0000	13.3.2.13/13-24
0x0_8560	PIWAR1—PCI inbound window attributes register 1	R/W	0x0000_0000	13.3.2.14/13-24
0x0_8568	PITAR0—PCI inbound translation address register 0	R/W	0x0000_0000	13.3.2.11/13-23
0x0_856C	Reserved, should be cleared	—	—	—
0x0_8570	PIBAR0—PCI inbound base address register 0	R/W	0x0000_0000	13.3.2.12/13-23
0x0_8578	PIWAR0—PCI inbound window attributes register 0	R/W	0x0000_0000	13.3.2.13/13-24
0x0_857C–0x0_CFFF	Reserved	—	—	—
Secondary DDR Memory Controller Memory Map				
0x0_D000	CS0_BNDS—Chip select memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x0_D008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	
0x0_D010–0x0_D028	Reserved	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_D080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x0_D084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	
0x0_D088–0x0_D0FF	Reserved	—	—	—
0x0_D100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-14
0x0_D104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x0_D108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-17
0x0_D10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-19
0x0_D110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21
0x0_D114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-23
0x0_D118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-25
0x0_D11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-26
0x0_D120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-26
0x0_D124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-29
0x0_D128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-30
0x0_D130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30
0x0_D140	Reserved	—	—	—
0x0_D148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-31
0x0_DBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x0002_0200	9.4.1.16/9-31
0x0_DBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x0000_0000	9.4.1.17/9-32
0x0_DE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.18/9-32
0x0_DE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.19/9-33
0x0_DE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.20/9-33
0x0_DE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.21/9-34
0x0_DE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.22/9-34
0x0_DE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.23/9-35
0x0_DE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.24/9-35
0x0_DE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.25/9-36
0x0_DE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.26/9-37
0x0_DE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.27/9-38
0x0_DE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.28/9-39
0x0_DE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.29/9-39
0x0_DE5C–0x0_DFFF	Reserved	—	—	—

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_E000–0x2_5FFF	Reserved	—	—	—
0x2_4000–0x2_400C	Reserved, should be cleared	—	—	—
Security Engine Address Map Registers				
Controller Registers				
0x3_0000–0x3_0FFF	Reserved, should be cleared	—	—	—
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	14.7.2.1/14-94
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	14.7.2.2/14-96
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	14.7.2.3/14-96
0x3_1020	ID—Identification register	R	0x0000_0000_0000_0040	14.7.2.4/14-98
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	14.7.2/14-93
0x3_1030	MCR—Master control register	R/W	0000_0000_0000_0000	14.7.2.6/14-99
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1180–0x3_11BF	DB*n*—Crypto-channel 1 descriptor buffers [0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1280–0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1380–0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1480–0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	14.5.2.1/14-35
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	14.5.2.2/14-36
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000_0000_0000	14.5.2.3/14-36
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000_0000_0000	14.5.2.4/14-37
0x3_2028	DEUSR—DEU status register	R	0x0000_0000_0000_0000	14.5.2.5/14-37
0x3_2030	DEUISR—DEU interrupt status register	R	0x0000_0000_0000_0000	14.5.2.6/14-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	14.5.2.7/14-40
0x3_2050	DEUEUG—DEU EU-Go register	W	0x0000_0000_0000_0000	14.5.2.8/14-41

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000 _0000_0000	14.5.2.9/14-42
0x3_2400	DEUK1—DEU key 1 register	W	—	14.5.2.10/14-42
0x3_2408	DEUK2—DEU key 2 register	W	—	14.5.2.10/14-42
0x3_2410	DEUK3—DEU key 3 register	W	—	14.5.2.10/14-42
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000 _0000_0000	14.5.2.11/14-42
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000 _0000_0000	14.5.6.1/14-68
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000 _0000_0000	14.5.6.2/14-71
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000 _0000_0000	14.5.6.3/14-71
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000 _0000_0000	14.5.6.4/14-72
0x3_4028	AESUSR—AESU status register	R	0x0000_0000 _0000_0000	14.5.6.5/14-73
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000 _0000_0000	14.5.6.6/14-74
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000 _0000_1000	14.5.6.7/14-75
0x3_4050	AESUEMR—AESU end-of-message register	W	0x0000_0000 _0000_0000	14.5.6.8/14-76
0x3_4100	AESU context memory registers	R/W	0x0000_0000 _0000_0000	14.5.6.9/14-77
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000 _0000_0000	14.5.6.9.5/14-81
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000 _0000_0000	14.5.6.9.6/14-81
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000 _0000_0000	14.5.4.1/14-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000 _0000_0000	14.5.4.3/14-55
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000 _0000_0000	14.5.4.4/14-56
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000 _0000_0000	14.5.4.5/14-56
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000 _0000_0000	14.5.4.6/14-57

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000 _0000_0000	14.5.4.7/14-58
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000 _0000_1000	14.5.4.8/14-59
0x3_6050	MDEUEUG—MDEU EU-Go register	W	0x0000_0000 _0000_0000	14.5.4.10/14-61
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000 _0000_0000	14.5.4.11/14-61
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000 _0000_0000	14.5.4.12/14-62
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000 _0000_0000	14.5.4.13/14-63
ARC Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000 _0000_0000	14.5.3.1/14-43
0x3_808	AFEUKSR—AFEU key size register	R/W	0x0000_0000 _0000_0000	14.5.3.3/14-44
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000 _0000_0000	14.5.3.4/14-45
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000 _0000_0000	14.5.3.5/14-46
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000 _0000_0000	14.5.3.6/14-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000 _0000_0000	14.5.3.7/14-47
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000 _0000_1000	14.5.3.8/14-49
0x3_8050	AFEUEMR—AFEU end of message register	W	0x0000_0000 _0000_0000	14.5.3.9/14-50
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000 _0000_0000	14.5.3.10.1/14-50
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000 _0000_0000	14.5.3.10.2/14-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	14.5.3.11/14-51
0x3_848	AFEUK1—AFEU key register 1	W	—	14.5.3.11/14-51
0x3_8800–0x3_8FFF	AFEU FIFO	R/W	0x0000_0000 _0000_0000	14.5.3.11.1/14-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000 _0000_0000	14.5.5.1/14-63
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000 _0000_0000	14.5.5.2/14-64

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000 _0000_0000	14.5.5.3/14-65
0x3_A028	RNGSR—RNG status register	R	0x0000_0000 _0000_0000	14.5.5.4/14-65
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000 _0000_0000	14.5.5.5/14-66
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000 _0000_1000	14.5.5.6/14-67
0x3_A050	RNGEUG—RNG EU-Go register	W	0x0000_0000 _0000_0000	14.5.5.7/14-68
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000 _0000_0000	14.5.5.8/14-68
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000 _0000_0000	14.5.1.1/14-26
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000 _0000_0000	14.5.1.2/14-28
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000 _0000_0000	14.5.1.3/14-28
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000 _0000_0000	14.5.1.5/14-29
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000 _0000_0000	14.5.1.6/14-30
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000 _0000_0000	14.5.1.7/14-31
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000 _0000_1000	14.5.1.8/14-32
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000 _0000_0000	14.5.1.3/14-28
0x3_C050	PKEUEUG—PKEU EU-Go	W	0x0000_0000 _0000_0000	14.5.1.9/14-33

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	14.5.1.10/14-34
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	
0x3_C340–0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_0000_0000	
0x3_C380–0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_0000_0000	
0x3_C3C0–0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_0000_0000	
0x3_C400–0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_0000_0000	
0x3_C800–0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_0000_0000	
QUICC Engine 2.0 Block				
0x10_0000–0x1F_FFFF	See Table 2-3 and Table 2-4 for more information.			

1 Depends on the reset configuration word high values. See Section 5.2.4.3.1, “LBLAWBAR0[BASE_ADDR] Reset Value,” for details.

2 Depends on the reset configuration word high values. See Section 5.2.4.4.1, “LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value,” for details.

3 Depends on the reset configuration word high values. See Section 5.2.4.5.1, “PCILAWBAR0[BASE_ADDR] Reset Value,” for details.

4 Depends on the reset configuration word high values. See Section 5.2.4.6.1, “PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value,” for details.

5 Depends on the reset configuration word high values. See Section 5.2.4.7.1, “DDRLAWBAR0[BASE_ADDR] Reset Value,” for details.

6 Depends on the reset configuration word high values. See Section 5.2.4.8.1, “DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value,” for details.

7 Depends on the reset configuration word high values.

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

2.4 QUICC Engine Internal Memory Map

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

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

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

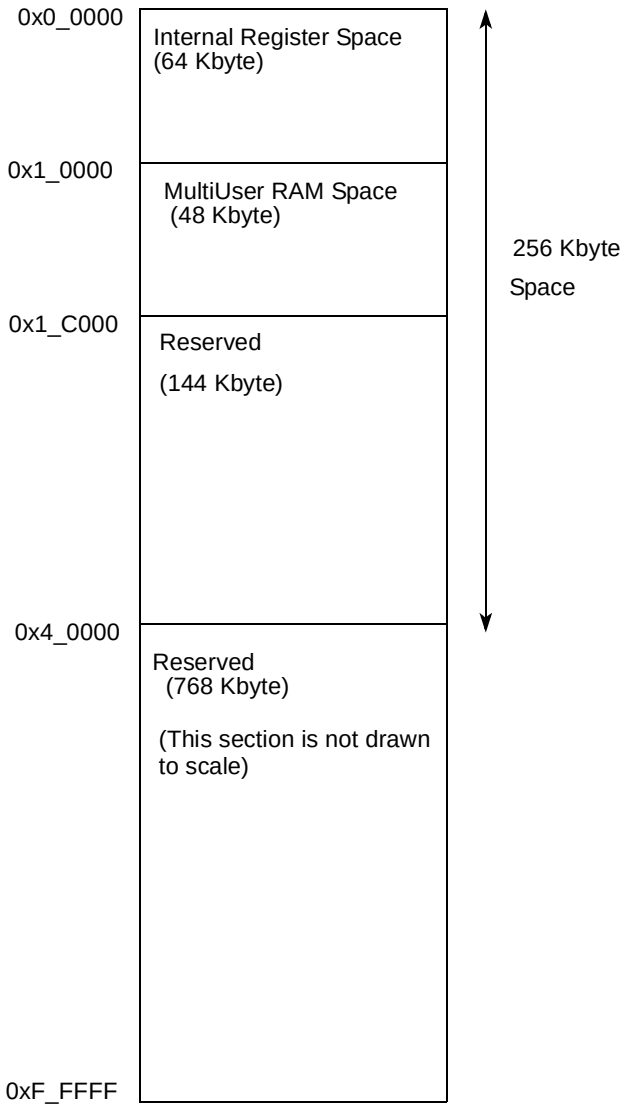


Figure 2-1. QUICC Engine 2.0 High-Level Memory Map

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

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

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

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

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

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	20.3.5/20-12	
0x0_0004	IDATA	I-RAM data register	4 bytes	20.3.6/20-13	
0x0_0008–0x0_007F	Reserved	—	120 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
Interrupt Controller					
0x0_0080	CICR	QUICC Engine system interrupt configuration register	4 bytes	19.3.1/19-23	
0x0_0084	CIVEC	QUICC Engine system interrupt vector register	4 bytes	19.3.12/19-34	
0x0_0088	CRIPNR	QUICC Engine RISC interrupt pending register	4 bytes	19.3.12/19-34	
0x0_008C	CIPNR	QUICC Engine system interrupt pending register	4 bytes	19.3.10/19-32	
0x0_0090	CIPXCC	QUICC Engine interrupt priority register	4 bytes	19.3.3/19-26	
0x0_0094	CIPYCC	QUICC Engine interrupt priority register	4 bytes	19.3.5/19-28	
0x0_0098	CIPWCC	QUICC Engine interrupt priority register	4 bytes	19.3.6/19-29	
0x0_009C	CIPZCC	QUICC Engine interrupt priority register	4 bytes	19.3.7/19-30	
0x0_00A0	CIMR	QUICC Engine system interrupt mask register	4 bytes	19.3.11/19-33	
0x0_00A4	CRIMR	QUICC Engine RISC interrupt mask register	4 bytes	19.3.11/19-33	
0x0_00A8	CICNR	QUICC Engine system interrupt control register	4 bytes	19.3.2/19-25	
0x0_00B0	CIPRTA	QUICC Engine system interrupt priority register for RISC tasks A	4 bytes	19.3.8/19-31	
0x0_00B4	CIPRTB	QUICC Engine system interrupt priority register for RISC tasks B	4 bytes	19.3.9/19-31	
0x0_00B8–0x0_00BB	Reserved	—	4 bytes		
0x0_00BC	CRICR	QUICC Engine system RISC interrupt control register	4 bytes	19.3.3/19-26	
0x0_00C0–0x0_00DC	Reserved	—	32 bytes		
0x0_00E0	CHIVEC	QUICC Engine high system interrupt vector register	4 bytes	19.3.15/19-37	
0x0_00E4–0x0_00FF	Reserved	—	28 bytes		
Communications Processor					
0x0_0100	CECR	QUICC Engine command register	4 bytes	20.3.1/20-2	Hard reset
0x0_0104	CECCR	QUICC Engine controller configuration register	4 bytes	20.3.8/20-14	Hard reset

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0108	CECDR	QUICC Engine command data register	4 bytes	20.3.2/20-10	
0x0_010C–0x0_0115	Reserved	—	10 bytes		
0x0_0116	CETER	QUICC Engine timer event register	2 bytes	20.4.4/20-20	
0x0_0118	Reserved	—	2 bytes		
0x0_011A	CETMR	QUICC Engine timers mask register	2 bytes	20.4.4/20-20	
0x0_011C	CETSCR	QUICC Engine time-stamp timer control register	4 bytes	20.3.9/20-16	
0x0_0120	CETSR1	QUICC Engine time-stamp register 1	4 bytes	20.3.9/20-16	
0x0_0124	CETSR2	QUICC Engine time-stamp register 2	4 bytes	20.3.9/20-16	
0x0_0128–0x0_012F	Reserved	—	8 bytes		
0x0_0130	CEVTER	QUICC Engine virtual tasks event register	4 bytes	20.3.3/20-11	
0x0_0134	CEVTMR	QUICC Engine virtual tasks mask register	4 bytes	20.3.3/20-11	
0x0_0138	CERCR	QUICC Engine RAM control register	2 bytes	20.3.4/20-12	
0x0_013C–0x0_015F	Reserved	—	36 bytes		
0x0_0160	CEEXE1	QUICC Engine external request 1 event register	2 bytes	20.5.1/20-22	
0x0_0162–0x0_0163	Reserved	—	2 bytes		
0x0_0164	CEEXM1	QUICC Engine external request 1 mask register	2 bytes	20.5.1/20-22	
0x0_0166–0x0_0167	Reserved	—	2 bytes		
0x0_0168	CEEXE2	QUICC Engine external request 2 event register	2 bytes	20.5.1/20-22	
0x0_016A–0x0_016B	Reserved	—	2 bytes		
0x0_016C	CEEXM2	QUICC Engine external request 2 mask register	2 bytes	20.5.1/20-22	
0x0_016E–0x0_016F	Reserved	—	2 bytes		
0x0_0170	CEEXE3	QUICC Engine external request 3 event register	2 bytes	20.5.1/20-22	
0x0_0172–0x0_0173	Reserved	—	2 bytes		
0x0_0174	CEEXM3	QUICC Engine external request 3 mask register	2 bytes	20.5.1/20-22	
0x0_0176–0x0_0177	Reserved	—	2 bytes		
0x0_0178	CEEXE4	QUICC Engine external request 4 event register	2 bytes	20.5.1/20-22	
0x0_017A–0x0_017B	Reserved	—	2 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_017C	CEEXM4	QUICC Engine external request 4 mask register	2 bytes	20.5.1/20-22	
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					
0x0_0400	CMXGCR	CMX general clock route register	4 bytes		Hard reset
0x0_0404	CMXSI1CR_L	CMX SI1 clock route low register	4 bytes		Hard reset
0x0_0408	CMXSI1CR_H	CMX SI1 clock route high register	4 bytes		Hard reset
0x0_040C	CMXSI1SYR	CMX SI1 SYNC route register	4 bytes		Hard reset
0x0_0410	CMXUCR1	CMX UCC1, UCC3 clock route register	4 bytes		Hard reset
0x0_0414	CMXUCR2	CMX UCC5, UCC7 clock route register	4 bytes		Hard reset
0x0_0418	CMXUCR3	CMX UCC2, UCC4 clock route register	4 bytes		Hard reset
0x0_041C	CMXUCR4	CMX UCC6, UCC8 clock route register	4 bytes		Hard reset
0x0_0420	CMXUPCR	CMX UPC clock route register	4 bytes		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		
0x0_0441	Reserved	—	3 bytes		
0x0_0444	GTCFR2	Timer 3 and timer 4 global configuration register	1 byte		
0x0_0445–0x0_044F	Reserved	—	11 bytes		
0x0_0450	GTMDR1	Timer 1 mode register	2 bytes		
0x0_0452	GTMDR2	Timer 2 mode register	2 bytes		
0x0_0454	GTRFR1	Timer 1 reference register	2 bytes		
0x0_0456	GTRFR2	Timer 2 reference register	2 bytes		
0x0_0458	GTCPR1	Timer 1 capture register	2 bytes		
0x0_045A	GTCPR2	Timer 2 capture register	2 bytes		
0x0_045C	GTCNR1	Timer 1 counter	2 bytes		
0x0_045E	GTCNR2	Timer 2 counter	2 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0460	GTMDR3	Timer 3 mode register	2 bytes		
0x0_0462	GTMDR4	Timer 4 mode register	2 bytes		
0x0_0464	GTRFR3	Timer 3 reference register	2 bytes		
0x0_0466	GTRFR4	Timer 4 reference register	2 bytes		
0x0_0468	GTCPR3	Timer 3 capture register	2 bytes		
0x0_046A	GTCPR4	Timer 4 capture register	2 bytes		
0x0_046C	GTCNR3	Timer 3 counter	2 bytes		
0x0_046E	GTCNR4	Timer 4 counter	2 bytes		
0x0_0470	GTEVR1	Timer 1 event register	2 bytes		
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		
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		
SPIn_BASE+0x24	Reserved	—	2 bytes		
SPIn_BASE+0x26	SPIE	SPI event register	1 byte		
SPIn_BASE+0x27	Reserved	—	3 bytes		
SPIn_BASE+0x2A	SPIM	SPI mask register	1 byte		
SPIn_BASE+0x2B	Reserved	—	2 bytes		
SPIn_BASE+0x2D	SPCOM	SPI command register	1 byte		
SPIn_BASE+0x2E	Reserved	—	2 bytes		
SPIn_BASE+0x30	SPITD	SPI transmit data register (cpu mode)	4 bytes		
SPIn_BASE+0x34	SPIRD	SPI receive data register (cpu mode)	4 bytes		
SPIn_BASE+0x38–SPIn_BASE+0x3F	Reserved	—	8 bytes		
MCC Registers					

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_0540	MCCE	MCC event register	4 bytes	34.2.6.1/34-38	
0x0_0544	MCCM	MCC mask register	4 bytes	34.2.6.1/34-38	
0x0_0548	MCCF	MCC configuration register	4 bytes	34.2.5/34-35	
0x0_054C	MERL	MCC emergency request level register	4 bytes	34.2.8/34-45	
0x0_0550–0x0_057F	Reserved	—	48 bytes		
0x0_0580–0x0_063F	Reserved	—	192 bytes		
BRG BRG_BASE = 0x00640					
BRG_BASE+0x00	BRGC1	BRG1 configuration register	4 bytes		
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	BRGC12	BRG12 configuration register	4 bytes		
BRG_BASE+0x30	BRGC13	BRG13 configuration register	4 bytes		
BRG_BASE+0x34	BRGC14	BRG14 configuration register	4 bytes		
BRG_BASE+0x38	BRGC15	BRG15 configuration register	4 bytes		
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	44.4.7.1/44-16	
0x0_06C1	USADD	USB address register	1 byte	44.4.7.2/44-17	
0x0_06C2	USCOM	USB command register	1 byte	44.4.7.4/44-19	
0x0_06C3	Reserved	—	1 byte		
0x0_06C4	USEP0	USB endpoint register 0	2 bytes	44.4.7.3/44-18	
0x0_06C6	USEP1	USB endpoint register 1	2 bytes	44.4.7.3/44-18	

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_06C8	USEP2	USB endpoint register 2	2 bytes	44.4.7.3/44-18	
0x0_06CA	USEP3	USB endpoint register 3	2 bytes	44.4.7.3/44-18	
0x0_06CC–0x0_06CF	Reserved	—	4 bytes		
0x0_06D0	USBER	USB event register	2 bytes	44.4.7.5/44-20	
0x0_06D2–0x0_06D3	Reserved	—	2 bytes		
0x0_06D4	USBMR	USB mask register	2 bytes	44.4.7.6/44-21	
0x0_06D6	Reserved	—	1 byte		
0x0_06D7	USBS	USB status register	1 byte	44.4.7.7/44-21	
0x0_06D8	USSFT	USB start of frame timer	4 bytes	44.4.7.8/44-22	
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		
SI1_BASE+0x02	SIBMR1	SI1 TDMB mode register	2 bytes		
SI1_BASE+0x04	SICMR1	SI1 TDMC mode register	2 bytes		
SI1_BASE+0x06	SIDMR1	SI1 TDMD mode register	2 bytes		
SI1_BASE+0x08	SIGLMR1_H	SI1 global mode register high	1 byte		
SI1_BASE+0x09	Reserved	Must be cleared	1 byte		
SI1_BASE+0x0A	SICMDR1_H	SI1 command register high	1 byte		
SI1_BASE+0x0B	Reserved	Must be cleared	1 byte		
SI1_BASE+0x0C	SISTR1_H	SI1 status register high	1 byte		
SI1_BASE+0x0D	Reserved	Must be cleared	1 byte		
SI1_BASE+0x0E	SIRSR1_H	SI1 RAM shadow address register high	2 bytes		
SI1_BASE+0x10	SITARC1	SI1 RAM counter Tx TDMA	1 byte		
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		
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		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
SI1_BASE+0x18–0x1F	Reserved	—	8 bytes		
SI1_BASE+0x20	SIEMR1	SI1 TDME mode register	2 bytes		
SI1_BASE+0x22	SIFMR1	SI1 TDMF mode register	2 bytes		
SI1_BASE+0x24	SIGMR1	SI1 TDMG mode register	2 bytes		
SI1_BASE+0x26	SIHMR1	SI1 TDMH mode register	2 bytes		
SI1_BASE+0x28	SIGLMR1_L	SI1 global mode register low	1 byte		
SI1_BASE+0x29	Reserved	Must be cleared	1 byte		
SI1_BASE+0x2A	SICMDR1_L	SI1 command register low	1 byte		
SI1_BASE+0x2B	Reserved	Must be cleared	1 byte		
SI1_BASE+0x2C	SISTR1_L	SI1 status register low	1 byte		
SI1_BASE+0x2D	Reserved	Must be cleared	1 byte		
SI1_BASE+0x2E	SIRSR1_L	SI1 RAM shadow address register low	2 bytes		
SI1_BASE+0x30	SITERC1	SI1 RAM counter Tx TDME	1 byte		
SI1_BASE+0x31	SITFRC1	SI1 RAM counter Tx TDMF	1 byte		
SI1_BASE+0x32	SITGRC1	SI1 RAM counter Tx TDMG	1 byte		
SI1_BASE+0x33	SITHRC1	SI1 RAM counter Tx TDMH	1 byte		
SI1_BASE+0x38	SIRERC1	SI1 RAM counter Rx TDME	1 byte		
SI1_BASE+0x39	SIRFRC1	SI1 RAM counter Rx TDMF	1 byte		
SI1_BASE+0x3A	SIRGRC1	SI1 RAM counter Rx TDMG	1 byte		
SI1_BASE+0x3B	SIRHRC1	SI1 RAM counter Rx TDMH	1 byte		
SI1_BASE+0x38–0x3F	Reserved	—	8 bytes		
SI1_BASE+0x40	SIML1	SI1 multiframe limit register	4 bytes		
SI1_BASE+0x44	SIENS1	SI1 extended diagnostic mode register	1 byte		
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		
0x0_1400–0x0_17FF	SIRxRAM	SI1 Rx routing table	1 Kbyte		
0x0_1800–0x0_1FFF	Reserved	—	2 Kbytes		
UCCx: x=1,2,3,4,5,6,7,8					

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC1_BASE=0x0_2000; UCC3_BASE=0x0_2200; UCC5_BASE=0x0_2400; UCC7_BASE=0x0_2600; UCC2_BASE=0x0_3000; UCC4_BASE=0x0_3200; UCC6_BASE=0x0_3400; UCC8_BASE=0x0_3600;					
Slow Mode (UART, BISYNC, QMC)					
UCC*x*_BASE+0x0	GUMR_Lx	UCCx general mode register (low)	4 bytes		
UCC*x*_BASE+0x4	GUMR_Hx	UCCx general mode register (high)	4 bytes		
UCC*x*_BASE+0x8	UPSMRx	UCCx protocol-specific mode register	2 bytes	25.3.4/25-8 26.3.2.5/26-9	
UCC*x*_BASE+0xA	Reserved	—	2 bytes		
UCC*x*_BASE+0xC	UTODRx	UCCx transmit on demand register	2 bytes	23.3.4/23-7	
UCC*x*_BASE+0xE	UDSRx	UCCx data synchronization register	2 bytes		
UCC*x*_BASE+0x10	UCCEx	UCCx event register	2 bytes	25.3.7/25-14 26.3.2.8/26-14 42.5.2/42-30	
UCC*x*_BASE+0x12	Reserved	—	2 bytes		
UCC*x*_BASE+0x14	UCCMx	UCCx mask register	2 bytes	25.3.7/25-14 26.3.2.8/26-14 42.5.2/42-30	
UCC*x*_BASE+0x16	Reserved	—	1 byte		
UCC*x*_BASE+0x17	UCCSx	UCCx status register	1 byte	25.3.8/25-16	
UCC*x*_BASE+0x18 –UCC*x*_BASE+0x1FF	Reserved	—			
Fast Mode (Ethernet, ATM, POS, HDLC, Transparent)					
UCC*x*_BASE+0x0	GUMRx	UCCx general mode register	4 bytes	27.4.2.1/27-6	
UCC*x*_BASE+0x4	UPSMRx	UCCx protocol-specific mode register	4 bytes	29.2.2.2/29-7 28.4.2.2/28-6	
UCC*x*_BASE+0x8	UTODRx	UCCx transmit on demand register	2 bytes	23.3.4/23-7	
UCC*x*_BASE+0xA	Reserved	—	2 bytes		
UCC*x*_BASE+0xC	UDSRx	UCCx data synchronization register	2 bytes	27.4.5/27-11	
UCC*x*_BASE+0xE	Reserved	—	2 bytes		
UCC*x*_BASE+0x10	UCCEx	UCCx event register	4 bytes	29.2.2.5/29-12 40.10.2/40-45 28.4.2.5/28-10	
UCC*x*_BASE+0x14	UCCMx	UCCx mask register	4 bytes	29.2.2.5/29-12 40.10.2/40-45 28.4.2.5/28-10	
UCC*x*_BASE+0x18	UCCSx	UCCx status register	1 byte	29.2.2.6/29-15	
UCC*x*_BASE+0x19 –UCC*x*_BASE+0x1F	Reserved	—	7 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC*x*_BASE+0*x*20	URFBx	UCC receive FIFO base	4 bytes	27.5.1/27-13	
UCC*x*_BASE+0*x*24	URFSx	UCC receive FIFO size	2 bytes	27.5.2/27-13	
UCC*x*_BASE+0*x*26	Reserved	—	2 bytes		
UCC*x*_BASE+0*x*28	URFETx	UCC receive FIFO emergency threshold	2 bytes	27.5.3/27-14	
UCC*x*_BASE+0*x*2A	URFSETx	UCC receive FIFO special emergency threshold	2 bytes	27.5.4/27-14	
UCC*x*_BASE+0*x*2C	UTFBx	UCC transmit FIFO base	4 bytes	27.5.5/27-15	
UCC*x*_BASE+0*x*30	UTFSx	UCC transmit FIFO size	2 bytes	27.5.6/27-15	
UCC*x*_BASE+0*x*32	Reserved	—	2 bytes		
UCC*x*_BASE+0x34	UTFETx	UCC transmit FIFO emergency threshold	2 bytes	27.5.7/27-16	
UCC*x*_BASE+0x36	Reserved	—	2 bytes		
UCC*x*_BASE+0x38	UTFTT	UCC transmit FIFO transmit threshold	2 bytes	27.5.8/27-16	
UCC*x*_BASE+0x3A	Reserved	—	2 bytes		
UCC*x*_BASE+0x3C	UTPTx	UCC transmit polling timer	2 bytes	27.4.3/27-10	
UCCx_BASE+0x3E	Reserved	—	2 bytes		
UCC*x*_BASE+0x40	URTRYx	UCC retry counter register	4 bytes	27.5.9/27-17	
UCC*x*_BASE+0x44 –UCC*x*_BASE+0x8F	Reserved	—	76 bytes		
UCC*x*_BASE+0x90	GUEMRx	UCC general extended mode register	1 byte	23.3.2/23-5	
UCCx_BASE+0x91 –UCCx_BASE+0xFF	Reserved	—	111 bytes		
Ethernet General Configuration					
UCC*x*_BASE+0x100	MACCFG1	Mac configuration register #1	4 bytes		
UCC*x*_BASE+0x104	MACCFG2	Mac configuration register #2	4 bytes		
UCC*x*_BASE+0x108	IPGIFG	Interframe gap register	4 bytes		
UCC*x*_BASE+0x10C	HAFDUP	Half-duplex register	4 bytes		
UCC*x*_BASE+0x110	Reserved	—	12 bytes		
UCC*x*_BASE+0x11C	EMTR	Ethernet MAC test register	4 bytes		
UCC*x*_BASE+0x120	MIIMCFG	MII mgmt configuration register	4 bytes		
UCC*x*_BASE+0x124	MIIMCOM	MII mgmt command register	4 bytes		
UCC*x*_BASE+0x128	MIIMADD	MII mgmt address register	4 bytes		
UCC*x*_BASE+0x12C	MIIMCON	MII mgmt control register	4 bytes		
UCC*x*_BASE+0x130	MIIMSTAT	MII mgmt status register	4 bytes		Read only

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UCC*x*_BASE+0x134	MIIMIND	MII mgmt indication register	4 bytes		Read only
UCC*x*_BASE+0x138	IFCTL	Interface control register	4 bytes		
UCC*x*_BASE+0x13C	IFSTAT	Interface status register	4 bytes		Read only
UCC*x*_BASE+0x140	MACSTNADDR1	Station address part 1 register	4 bytes		
UCCx_BASE+0x144	MACSTNADDR2	Station address part 2 register	4 bytes		
UCC*x*_BASE+0x148	Reserved	—	8 bytes		
UCC*x*_BASE+0x150	UEMPR	UCC Ethernet MAC parameter register	4 bytes		
UCC*x*_BASE+0x154	UTBIPAR	UCC TBI address	4 bytes		
UCC*x*_BASE+0x158	UESCR	UCC Ethernet statistics control register	2 bytes		
UCC*x*_BASE+0x15A–0x17F	Reserved	—			
Ethernet Statistics Counters					
UCCx_BASE+0x180	TX64	Transmit 64-byte frame counter	4 bytes		
UCCx_BASE+0x184	TX127	Transmit 65- to 127-byte frame counter	4 bytes		
UCCx_BASE+0x188	TX255	Transmit 128- to 255-byte frame counter	4 bytes		
UCCx_BASE+0x18C	RX64	Receive 64-byte frame counter	4 bytes		
UCCx_BASE+0x190	RX127	Receive 65- to 127-byte frame counter	4 bytes		
UCC*x*_BASE+0x194	RX255	Receive 128- to 255-byte frame counter	4 bytes		
UCC*x*_BASE+0x198	TXOK	Transmit good bytes counter	4 bytes		
UCC*x*_BASE+0x19C	TXCF	Transmit control frame counter	4 bytes		
UCC*x*_BASE+0x1A0	TMCA	Transmit multicast control frame counter	4 bytes		
UCC*x*_BASE+0x1A4	TBCA	Transmit broadcast packet counter	4 bytes		
UCC*x*_BASE+0x1A8	RXFOK	Receive frame OK counter	4 bytes		
UCC*x*_BASE+0x1B0	RBYT	Receive good and bad bytes counter	4 bytes		
UCC*x*_BASE+0x1AC	RXBOK	Receive bytes OK counter	4 bytes		
UCC*x*_BASE+0x1B4	RMCA	Receive multicast packet counter	4 bytes		
UCC*x*_BASE+0x1B8	RBCA	Receive broadcast packet counter	4 bytes		
UCC*x*_BASE+0x1BC	SCAR	Statistics carry register	4 bytes		
UCC*x*_BASE+0x1C0	SCAM	Statistics carry mask register	4 bytes		
UCC*x*_BASE+0x1C4	RxDiscOV	Overrun condition in Rx FIFO counter	4 bytes		
UCC*x*_BASE+0x1C8–0x1FF	Reserved	—	56 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UTOPIA POS Controllers UTOPIA/POS Controller 1: UPC1_BASE=0x0_2E00 UTOPIA/POS Controller 2: UPC2_BASE=0x0_3E00					
UPCn_BASE+0x00	UPGCR	UTOPIA/POS general configuration register	2 bytes		
UPCn_BASE+0x04	UPLPA	UTOPIA/POS last PHY address	2 bytes		
UPCn_BASE+0x08	UPHEC	ATM HEC register	2 bytes		
UPCn_BASE+0x0C	UPUC	UTOPIA/POS UCC configuration	4 bytes		
UPCn_BASE+0x10	UPDC1	UTOPIA/POS device 1 configuration	4 bytes		
UPCn_BASE+0x14	UPDC2	UTOPIA/POS device 2 configuration	4 bytes		
UPCn_BASE+0x18	UPDC3	UTOPIA/POS device 3 configuration	4 bytes		
UPCn_BASE+0x1C	UPDC4	UTOPIA/POS device 4 configuration	4 bytes		
UPCn_BASE+0x20	Reserved	—	1 byte		
UPCn_BASE+0x28	Reserved	—	4 bytes		
UPCn_BASE+0x30	UPDRS1_H	UTOPIA/POS device 1 rate select	4 bytes		
UPCn_BASE+0x34	UPDRS1_L	UTOPIA/POS device 1 rate select	4 bytes		
UPCn_BASE+0x38	UPDRS2_H	UTOPIA/POS device 2 rate select	4 bytes		
UPCn_BASE+0x3C	UPDRS2_L	UTOPIA/POS device 2 rate select	4 bytes		
UPCn_BASE+0x40	UPDRS3_H	UTOPIA/POS device 3 rate select	4 bytes		
UPCn_BASE+0x44	UPDRS3_L	UTOPIA/POS device 3 rate select	4 bytes		
UPCn_BASE+0x48	UPDRS4_H	UTOPIA/POS device 4 rate select	4 bytes		
UPCn_BASE+0x4C	UPDRS4_L	UTOPIA/POS device 4 rate select	4 bytes		
UPCn_BASE+0x50	UPDRP1	UTOPIA/POS device 1 receive priority low	4 bytes		
UPCn_BASE+0x54	UPDRP2	UTOPIA/POS device 2 receive priority low	4 bytes		
UPCn_BASE+0x58	UPDRP3	UTOPIA/POS device 3 receive priority low	4 bytes		
UPCn_BASE+0x5C	UPDRP4	UTOPIA/POS device 4 receive priority low	4 bytes		
UPCn_BASE+0x60	UPDE1	UTOPIA/POS device 1 event	4 bytes		
UPCn_BASE+0x64	UPDE2	UTOPIA/POS device 2 event	4 bytes		
UPCn_BASE+0x68	UPDE3	UTOPIA/POS device 3 event	4 bytes		
UPCn_BASE+0x6C	UPDE4	UTOPIA/POS device 4 event	4 bytes		
UPCn_BASE+0x70	UPRP1	UTOPIA/POS device 1 internal rate config	2 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
UPCn_BASE+0x72	UPRP2	UTOPIA/POS device 2 internal rate config	2 bytes		
UPCn_BASE+0x74	UPRP3	UTOPIA/POS device 3 internal rate config	2 bytes		
UPCn_BASE+0x76	UPRP4	UTOPIA/POS device 4 internal rate config	2 bytes		
UPCn_BASE+0x80	UPTIRR1_0	Device 1 transmit internal rate 0	2 bytes		
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	UPTIRR2_0	Device 2 transmit internal rate 0	2 bytes		
UPCn_BASE+0x8A	UPTIRR2_1	Device 2 transmit internal rate 1	2 bytes		
UPCn_BASE+0x8C	UPTIRR2_2	Device 2 transmit internal rate 2	2 bytes		
UPCn_BASE+0x8E	UPTIRR2_3	Device 2 transmit internal rate 3	2 bytes		
UPCn_BASE+0x90	UPTIRR3_0	Device 3 transmit internal rate 0	2 bytes		
UPCn_BASE+0x92	UPTIRR3_1	Device 3 transmit internal rate 1	2 bytes		
UPCn_BASE+0x94	UPTIRR3_2	Device 3 transmit internal rate 2	2 bytes		
UPCn_BASE+0x96	UPTIRR3_3	Device 3 transmit internal rate 3	2 bytes		
UPCn_BASE+0x98	UPTIRR4_0	Device 4 transmit internal rate 0	2 bytes		
UPCn_BASE+0x9A	UPTIRR4_1	Device 4 transmit internal rate 1	2 bytes		
UPCn_BASE+0x9C	UPTIRR4_2	Device 4 transmit internal rate 2	2 bytes		
UPCn_BASE+0x9E	UPTIRR4_3	Device 4 transmit internal rate 3	2 bytes		
UPCn_BASE+0xA0	UPER1	Device 1 port enable register	4 bytes		
UPCn_BASE+0xA4	UPER2	Device 2 port enable register	4 bytes		
UPCn_BASE+0xA8	UPER3	Device 3 port enable register	4 bytes		
UPCn_BASE+0xAC	UPER4	Device 4 port enable register	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	19.1.8.1/19-6	
0x0_4004	SDMR	Serial DMA mode register	4 bytes	19.1.8.2/19-7	
0x0_4008	SDTR1	SDMA system bus threshold register	4 bytes	19.1.8.3/19-9	

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_400C	SDTR2	SDMA secondary bus threshold register	4 bytes	19.1.8.3/19-9	
0x0_4010	SDHY1	SDMA system bus hysteresis register	4 bytes	19.1.8.4/19-11	
0x0_4014	SDHY2	SDMA secondary bus hysteresis register	4 bytes	19.1.8.4/19-11	
0x0_4018	SDTA1	SDMA system bus address register	4 bytes	19.1.8.5/19-12	
0x0_401C	SDTA2	SDMA secondary bus address register	4 bytes	19.1.8.5/19-12	
0x0_4020	SDTM1	SDMA system bus MSNUM register	4 bytes	19.1.8.6/19-12	
0x0_4024	SDTM2	SDMA secondary bus MSNUM register	4 bytes	19.1.8.6/19-12	
0x0_4028–0x0_4037	Reserved	—	16 bytes		
0x0_4038	SDAQR	SDMA address bus qualify register	4 bytes	19.1.8.7/19-13	
0x0_403C	SDAQMR	SDMA address bus qualify mask register	4 bytes	19.1.8.8/19-13	
0x0_4044	SDEBCR	SDMA CAM entries base register	4 bytes	19.1.8.9/19-14	
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_47FF	Reserved	—	640 bytes		
IEEE 1588					
IEEE 1588 Timer Mode Registers					
0x0_4800	TMR_CTRL		4 bytes		
0x0_4804	TMR_TEVENT		4 bytes		
0x0_4808	TMR_TEMASK		4 bytes		
0x0_480C	TMR_CNT_L		4 bytes		
0x0_4810	TMR_CNT_H		4 bytes		
0x0_4814	TMR_ADD		4 bytes		
0x0_4818	TMR_ACC		4 bytes		
0x0_481C	TMR_PRSC		4 bytes		
0x0_4820	TMROFF_L		4 bytes		
0x0_4824	TMROFF_H		4 bytes		
0x0_4828	TMR_ALARM1_L		4 bytes		
0x0_482C	TMR_ALARM1_H		4 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_4830	TMR_ALARM2_L		4 bytes		
0x0_4834	TMR_ALARM2_H		4 bytes		
0x0_4838	TMR_FIPER1		4 bytes		
0x0_483C	TMR_FIPER2		4 bytes		
0x0_4840	TMR_FIPER3		4 bytes		
0x0_4844	TMR_ETTS1_L		4 bytes		
0x0_4848	TMR_ETTS1_H		4 bytes		
0x0_484C	TMR_ETTS2_L		4 bytes		
0x0_4850	TMR_ETTS2_H		4 bytes		
IEEE 1588 Time Stamp Unit 1 Mode Registers					
0x0_4880	PTP1_TSPDR1		4 bytes		
0x0_4884	PTP1_TSPDR2		4 bytes		
0x0_4888	PTP1_TSPDR3		4 bytes		
0x0_488C	PTP1_TSPDR4		4 bytes		
0x0_4890	PTP1_TSPOV		4 bytes		
0x0_4894	PTP1_TSMR		4 bytes		
0x0_4898	PTP1_TMR_ PEVENT		4 bytes		
0x0_489C	PTP1_TMR_ PEMASK		4 bytes		
0x0_48A0	TMR_UC1_ RXTS_H		4 bytes		
0x0_48A4	TMR_UC3_ RXTS_H		4 bytes		
0x0_48A8	TMR_UC5_ RXTS_H		4 bytes		
0x0_48AC	TMR_UC7_ RXTS_H		4 bytes		
0x0_48B0	TMR_UC1_ RXTS_L		4 bytes		
0x0_48B4	TMR_UC3_ RXTS_L		4 bytes		
0x0_48B8	TMR_UC5_ RXTS_L		4 bytes		
0x0_48BC	TMR_UC7_ RXTS_L		4 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_48C0	TMR_UC1_ TXTS_H		4 bytes		
0x0_48C4	TMR_UC3_ TXTS_H		4 bytes		
0x0_48C8	TMR_UC5_ TXTS_H		4 bytes		
0x0_48CC	TMR_UC7_ TXTS_H		4 bytes		
0x0_48D0	TMR_UC1_ TXTS_L		4 bytes		
0x0_48D4	TMR_UC3_ TXTS_L		4 bytes		
0x0_48D8	TMR_UC5_ TXTS_L		4 bytes		
0x0_48DC	TMR_UC7_ TXTS_L		4 bytes		
IEEE 1588 Time Stamp Unit 1 Mode Registers					
0x0_4900	PTP2_TSPDR1	R/W	4 bytes		
0x0_4904	PTP2_TSPDR2	R/W	4 bytes		
0x0_4908	PTP2_TSPDR3	R/W	4 bytes		
0x0_490C	PTP2_TSPDR4	R/W	4 bytes		
0x0_4910	PTP2_TSPOV	R/W	4 bytes		
0x0_4914	PTP2_TSMR	R/W	4 bytes		
0x0_4918	PTP2_TMR_ PEVENT	R/W	4 bytes		
0x0_491C	PTP2_TMR_ PEMASK	R/W	4 bytes		
0x0_4920	TMR_UC2_ RXTS_H	R	4 bytes		
0x0_4924	TMR_UC4_ RXTS_H	R	4 bytes		
0x0_4928	TMR_UC6_ RXTS_H	R	4 bytes		
0x0_492C	TMR_UC8_ RXTS_H	R	4 bytes		
0x0_4930	TMR_UC2_ RXTS_L	R	4 bytes		
0x0_4934	TMR_UC4_ RXTS_L	R	4 bytes		

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

Internal Address	Abbreviation	Name	Size	Section/Page	Comments
0x0_4938	TMR_UC6_RXTS_L	R	4 bytes		
0x0_493C	TMR_UC8_RXTS_L	R	4 bytes		
0x0_4940	TMR_UC2_TXTS_H	R	4 bytes		
0x0_4944	TMR_UC4_TXTS_H	R	4 bytes		
0x0_4948	TMR_UC6_TXTS_H	R	4 bytes		
0x0_494C	TMR_UC8_TXTS_H	R	4 bytes		
0x0_4950	TMR_UC2_TXTS_L	R	4 bytes		
0x0_4954	TMR_UC4_TXTS_L	R	4 bytes		
0x0_4958	TMR_UC6_TXTS_L	R	4 bytes		
0x0_495C	TMR_UC8_TXTS_L	R	4 bytes		
0x0_4960–0x0_4FFF	Reserved	—	1696 bytes		
0x0_5000–0x0_7FFF	Reserved	—	12 Kbytes		
0x0_8000–0x3_FFFF	RAM Space				
0x0_8000–0x0_FFFF	Reserved		32 Kbytes		
Multiuser RAM					
0x1_0000–0x1_BFFF	MURAM	Multi-user RAM	48 Kbytes		
0x1_C000–0x3_FFFF	Reserved	—	144 Kbytes		
0x4_0000–0xF_FFFF	Reserved	—	768 Kbytes		

Chapter 3
Signal Descriptions

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

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

NOTE

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

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

3.1 Signals Overview

The MPC8360E signals are grouped as follows:

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

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

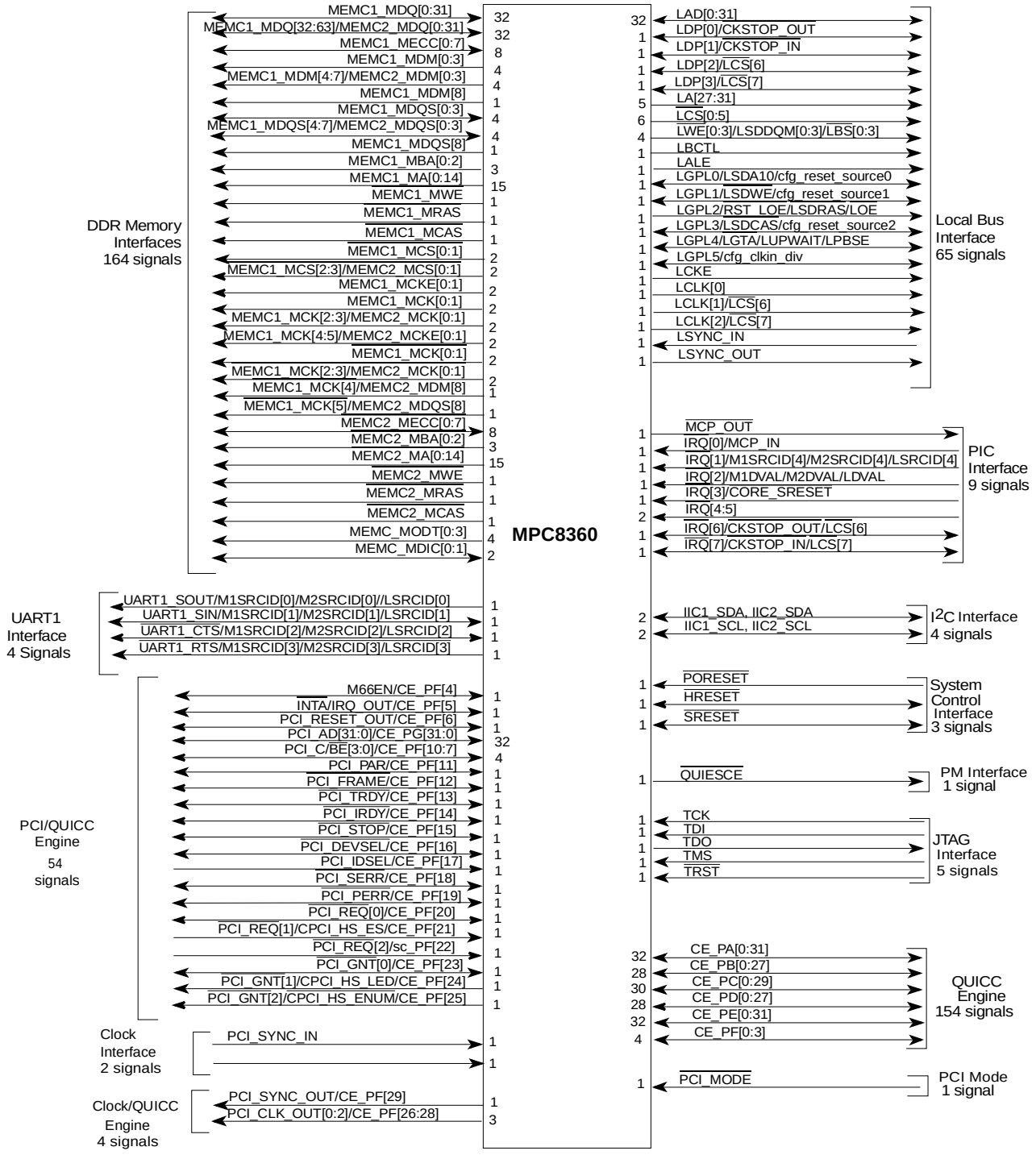


Figure 3-1. MPC8360E Signal Groupings

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

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

Table 3-1. MPC8360E Signal Reference by Functional Block

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
Primary DDR SDRAM Memory Controller Interface						
MEMC1_MDQ[0:31]	—	Primary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQ[32:63]/ MEMC2_MDQ[0:31]	MEMC1_MDQ[32:63]	Primary DDR data	DDR	32	I/O	9-3/9-6
	MEMC2_MDQ[0:31]	Secondary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MECC[0:4]	MEMC1_MSRCID[0:4]	Primary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC1_MECC[5]	MEMC1_MDVAL	Primary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC1_MECC[6:7]	—	Primary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC1_MDM[0:3]	—	Primary DDR data mask 0–3	DDR	4	O	9-3/9-6
MEMC1_MDM[4:7]/ MEMC2_MDM[0:3]	MEMC1_MDM[4:7]	Primary DDR data mask 4–7	DDR	4	O	9-3/9-6
	MEMC2_MDM[0:3]	Secondary DDR data mask 0–3			O	9-3/9-6
MEMC1_MDM[8]	—	Primary DDR ECC data mask	DDR	1	O	9-3/9-6
MEMC1_MDQS[0:3]	—	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
MEMC1_MDQS[4:7]/ MEMC2_MDQS[0:3]	MEMC1_MDQS[4:7]	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
	MEMC2_MDQS[0:3]	Secondary DDR data strobe			I/O	9-3/9-6
MEMC1_MDQS[8]	—	Primary DDR data strobe	DDR	1	I/O	9-3/9-6
MEMC1_MBA[0:2]	—	Primary DDR bank select	DDR	3	O	9-3/9-6
MEMC1_MA[0:14]	—	Primary DDR address	DDR	15	O	9-3/9-6
$\overline{\text{MEMC1\_MWE}}$	—	Primary DDR write enable	DDR	1	O	9-3/9-6
$\overline{\text{MEMC1\_MRAS}}$	—	Primary DDR row address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC1\_MCAS}}$	—	Primary DDR column address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC1\_MCS}}$[0:1]	—	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
$\overline{\text{MEMC1\_MCS}}$[2:3]/ $\overline{\text{MEMC2\_MCS}}$[0:1]	$\overline{\text{MEMC1\_MCS}}$[2:3]	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
	$\overline{\text{MEMC2\_MCS}}$[0:1]	Secondary DDR chip select (2/DIMM)			O	9-3/9-6
MEMC1_MCKE[0:1]	—	Primary DDR clock enable	DDR	2	O	9-4/9-9

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
MEMC1_MCK[0:1]	—	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
MEMC1_MCK[2:3]/ MEMC2_MCK[0:1]	MEMC1_MCK[2:3]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCK[0:1]	Secondary DDR differential clocks (positive)	DDR		O	9-4/9-9
MEMC1_MCK[4:5]/ MEMC2_MCKE[0:1]	MEMC1_MCK[4:5]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCKE[0:1]	Secondary DDR clock enable			O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[0:1]	—	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[2:3]/ $\overline{\text{MEMC2\_MCK}}$[0:1]	$\overline{\text{MEMC1\_MCK}}$[2:3]	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
	$\overline{\text{MEMC2\_MCK}}$[0:1]	Secondary DDR differential clocks (negative)	DDR		O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[4]/ MEMC2_MDM[8]	$\overline{\text{MEMC1\_MCK}}$[4]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDM[8]	Secondary DDR ECC data mask	DDR		O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[5]/ MEMC2_MDQS[8]	$\overline{\text{MEMC1\_MCK}}$[5]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDQS[8]	Secondary DDR data strobe	DDR		I/O	9-4/9-9
MEMC1_MODT[0:3]	—	On-die termination for DRAM chip	DDR	4	O	9-3/9-6
MEMC1_MDIC[0:1]	—	DRAM driver impedance calibration	DDR	2	I/O	9-3/9-6
Secondary DDR SDRAM Memory Controller Interface						
MEMC2_MECC[0:4]	MEMC2_MSRCID[0:4]	Secondary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC2_MECC[5]	MEMC2_MDVAL	Secondary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC2_MECC[6:7]	—	Secondary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC2_MBA[0:2]	—	Secondary DDR bank select	DDR	3	O	9-3/9-6
MEMC2_MA[0:14]	—	Secondary DDR address	DDR	15	O	9-3/9-6
$\overline{\text{MEMC2\_MWE}}$	—	Secondary DDR write enable	DDR	1	O	9-3/9-6
$\overline{\text{MEMC2\_MRAS}}$	—	Secondary DDR row address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC2\_MCAS}}$	—	Secondary DDR column address strobe	DDR	1	O	9-3/9-6

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI						
PCI_INTA/ IRQ_OUT/ CE_PF5	PCI_INTA	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
	CE_PF5	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_RESET_OUT/ CE_PF6	PCI_RESET_OUT	PCI reset	PCI	1	O	13-3/13-5
	CE_PF6	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_AD[31:0]/ CE_PG[31:0]	PCI_AD[31:0]	PCI address/data	PCI	32	I/O	13-3/13-5
	CE_PG[31:0]	QUICC Engine parallel I/O	PIO		I/O	3-18/3-61
PCI_C/BE[3:0]/ CE_PF[10:7]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	BE[3:0]	Byte enable	PCI		I/O	13-3/13-5
	CE_PF[10:7]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_PAR/ CE_PF11	PCI_PAR	PCI parity	PCI	1	I/O	13-3/13-5
	CE_PF11	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_FRAME/ CE_PF12	PCI_FRAME	PCI frame	PCI	1	I/O	13-3/13-5
	CE_PF12	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_TRDY/ CE_PF13	PCI_TRDY	PCI target ready	PCI	1	I/O	13-3/13-5
	CE_PF13	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_IRDY/ CE_PF14	PCI_IRDY	PCI initiator ready	PCI	1	I/O	13-3/13-5
	CE_PF14	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_STOP/ CE_PF15	PCI_STOP	PCI stop	PCI	1	I/O	13-3/13-5
	CE_PF15	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_DEVSEL/ CE_PF16	PCI_DEVSEL	PCI device select	PCI	1	I/O	13-3/13-5
	CE_PF16	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_IDSEL/ CE_PF17	PCI_IDSEL	PCI initial device select	PCI	1	I	13-3/13-5
	CE_PF17	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_SERR/ CE_PF18	PCI_SERR	PCI system error	PCI	1	I/O	13-3/13-5
	CE_PF18	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_PERR CE_PF19	PCI_PERR	PCI parity error	PCI	1	I/O	13-3/13-5
	CE_PF19	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_REQ0/ CE_PF20	PCI_REQ0	PCI request 0	PCI	1	I/O	13-3/13-5
	CE_PF20	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_REQ1/ CPCI_HS_ES/ CE_PF21	PCI_REQ1	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
	CE_PF21	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_REQ2/ CE_PF22	PCI_REQ2	PCI request 2	PCI	1	I	13-3/13-5
	CE_PF22	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_GNT0/ CE_PF23	PCI_GNT0	PCI grant 0	PCI	1	I/O	13-3/13-5
	CE_PF23	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_GNT1/ CPCI_HS_LED/ CE_PF24	PCI_GNT1	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
	CE_PF24	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_GNT2/ CPCI_HS_ENUM/ CE_PF25	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
	CE_PF25	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_MODE	—	PCI mode select	PCI	1	I	5-32/5-24
M66EN/CE_PF4	M66EN	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
	CE_PF4	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
Local Bus Controller Interface						
LAD[0:31]	—	LBC address/data	LBC	32	I/O	10-2/10-5
LDP[0]/ CKSTOP_OUT	LDP[0]	LBC data parity 0	LBC	1	I/O	10-2/10-5
	CKSTOP_OUT	Checkstop out	CPU		O	4-3/4-4
LDP[1]/ CKSTOP_IN	LDP[1]	LBC data parity 1	LBC	1	I/O	10-2/10-5
	CKSTOP_IN	Checkstop in	CPU		I	4-3/4-4
LDP[2]/LCS[6]	LDP[2]	LBC data parity 2	LBC	1	I/O	10-2/10-5
	LCS[6]	LBC chip select 6			O	10-2/10-5
LDP[3]/LCS[7]	LDP[3]	LBC data parity 3	LBC	1	I/O	10-2/10-5
	LCS[7]	LBC chip select 7			O	10-2/10-5
LA[27:31]	—	LBC port address 27–31	LBC	5	O	10-2/10-5
LCS[0:5]	—	LBC chip select 0–5	LBC	6	O	10-2/10-5
LWE[0:3]/ LSDDQM[0:3]/ LBS[0:3]	LWE[0:3]	LBC write enable 0–3	LBC	4	O	10-2/10-5
	LSDDQM[0:3]	Byte lane data mask 0–3			O	3-4/3-19
	LBS[0:3]	Byte select 0–3			O	10-2/10-5

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

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

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

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

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
UART1_SIN/ M1SRCID[1]/ M2SRCID[1] LSRCID[1]	UART1_SIN	UART1 serial data in	UART	1	I	16-2/16-3
	M1SRCID[1]	DDR memory debug source port ID 1	DDR		O	9-7/9-13
	M2SRCID[1]	Secondary DDR memory debug source port ID 1	DDR		O	9-7/9-13
	LSRCID[1]	LBC debug source port ID 1	LBC		O	10-2/10-5
$\overline{\text{UART1\_CTS}}$/ M1SRCID[2]/ M2SRCID[2] LSRCID[2]	$\overline{\text{UART1\_CTS}}$	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-13
	M2SRCID2	Secondary DDR memory debug source port ID 2	DDR		O	9-7/9-13
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
$\overline{\text{UART1\_RTS}}$ M1SRCID[3]/ M2SRCID[3] LSRCID[3]	$\overline{\text{UART1\_RTS}}$	UART1 ready to send	UART	1	O	16-2/16-3
	M1SRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-13
	M2SRCID3	Secondary DDR memory debug source port ID 3	DDR		O	9-7/9-13
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5
UART2 signals are multiplexed with QUICC Engine functions; see Table 3-16						
I^2C Interface						
IIC1_SDA	—	I2C serial data	I2C1	1	I/O	15-2/15-4
IIC1_SCL	—	I2C serial clock	I2C1	1	I/O	15-2/15-4
IIC2_SDA	—	I2C serial data	I2C2	1	I/O	15-2/15-4
IIC2_SCL/	—	I2C serial clock	I2C2	1	I/O	15-2/15-4
QUICC Engine Block						
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-12/3-30
CE_PB[0:27]	—	QUICC Engine parallel port B	QUICC Engine	28	I/O	3-13/3-35
CE_PC[0:29]	—	QUICC Engine parallel port C	QUICC Engine	30	I/O	3-14/3-41
CE_PD[0:27]	—	QUICC Engine parallel port D	QUICC Engine	28	I/O	3-15/3-45
CE_PE[0:31]	—	QUICC Engine parallel port E	QUICC Engine	32	I/O	3-16/3-50

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
CE_PF[0:3]	—	QUICC Engine parallel port F	QUICC Engine	4	I/O	3-17/3-56
Clocks						
PCI_CLK_OUT[0:2] CE_PF[26:28]	PCI_CLK_OUT[0:2]	PCI clock out 0–2	Clocks	3	O	4-2/4-3
	CE_PF[26:28]	QUICC Engine port F	QUICC Engine		I/O	3-17/3-56
CLK_IN	—	Clock input	Clocks	1	I	4-2/4-3
PCI_SYNC_IN/ PCI_CLOCK	PCI_SYNC_IN	PCI clock sync input	Clocks	1	I	4-2/4-3
	PCI_CLOCK	PCI clock	Clocks			
PCI_SYNC_OUT/ CE_PF[29]	PCI_SYNC_OUT	PCI clock sync output	Clocks	1	O	4-2/4-3
	CE_PF[29]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
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
$\overline{\text{TRST}}$	—	Test reset	JTAG	1	I	17-2/17-2
Test						
TEST_SEL (MPC8360) $\overline{\text{TEST\_SEL}}$ (MPC8358)	—	Test mode	TEST	1	I	—
TEST	—	Test mode	TEST	1	I	—
PMC						
$\overline{\text{QUIESCE}}$	—	Quiesce state	PMC	1	O	5-77/5-73
System Control						
$\overline{\text{PORESET}}$	—	Power-on reset	System control	1	I	4-1/4-1
$\overline{\text{HRESET}}$	—	Hard reset	System control	1	I/O	4-1/4-1
$\overline{\text{SRESET}}$	—	Soft reset	System control	1	I/O	4-1/4-1

Table 3-2 lists the signals in alphabetical order.

Table 3-2. MPC8360E Alphabetical Signal Reference

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
CE_PA[0:31]	—	QUICC Engine parallel port A	QUICC Engine	32	I/O	3-12/3-30
CE_PB[0:27]	—	QUICC Engine parallel port B	QUICC Engine	28	I/O	3-13/3-35
CE_PC[0:29]	—	QUICC Engine parallel port C	QUICC Engine	3032	I/O	3-14/3-41
CE_PD[0:27]	—	QUICC Engine parallel port D	QUICC Engine	28	I/O	3-15/3-45
CE_PE[0:31]	—	QUICC Engine parallel port E	QUICC Engine	32	I/O	3-16/3-50
CE_PF[0:3]	—	QUICC Engine parallel port F	QUICC Engine	4	I/O	3-17/3-56
CLK_IN	—	Clock input	Clocks	1	I	4-2/4-3
$\overline{\text{HRESET}}$	—	Hard reset	System control	1	I/O	4-4/4-5
IIC1_SCL	—	I2C serial clock	I2C1	1	I/O	15-2/15-4
IIC1_SDA	—	I2C serial data	I2C1	1	I/O	15-2/15-4
IIC2_SCL	—	I2C serial clock	I2C2	1	I/O	15-2/15-4
IIC2_SDA	—	I2C serial data	I2C2	1	I/O	15-2/15-4
$\overline{\text{IRQ}}$[0]/ $\overline{\text{MCP\_IN}}$	$\overline{\text{IRQ}}$[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]/ M1SRCID[4]/ M2SRCID[4]/ LSRCID[4]	$\overline{\text{IRQ}}$[1]	External interrupt 1	IPIC	1	I	8-2/8-5
	M1SRCID[4	DDR memory debug source port ID 4	DDR		O	9-7/9-13
	M2SRCID[4]	Secondary DDR memory debug source port ID 4	DDR		O	9-7/9-13
	LSRCID[4]	LBC debug source port ID 4	LBC		O	10-2/10-5
$\overline{\text{IRQ}}$[2]/ M1DVAL/ M2DVAL/ LDVAL	$\overline{\text{IRQ}}$[2]	External interrupt 2	IPIC	1	I	8-2/8-5
	M1DVAL	DDR memory debug data valid	DDR		O	9-7/9-13
	M2DVAL	Secondary DDR memory debug data valid	DDR		O	9-7/9-13
	LDVAL	LBC debug data valid	LBC		O	10-2/10-5
$\overline{\text{IRQ}}$[3]/ $\overline{\text{CORE\_SRESET}}$	$\overline{\text{IRQ}}$[3]	External interrupt 3	IPIC	1	I	8-2/8-5
	$\overline{\text{CORE\_SRESET}}$	Core soft reset	Core	1	I	
$\overline{\text{IRQ}}$[4:5]	—	External interrupt 4–5	IPIC	1	I	8-2/8-5

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
IRQ[6]/ LCS[6] CKSTOP_OUT	IRQ[6]	External interrupt 6	IPIC	1	I	8-2/8-5
	LCS[6]	LBC chip select 6	LBC		O	10-2/10-5
	CKSTOP_OUT	Checkstop output	CPU		O	4-5/4-9
IRQ[7]/ LCS[7]/ CKSTOP_IN	IRQ[7]	External interrupt 7	IPIC	1	I	8-2/8-5
	LCS[7]	LBC chip select 7	LBC		O	10-2/10-5
	CKSTOP_IN	Checkstop input	CPU		I	4-5/4-9
LA[27:31]	—	LBC port address 27–31	LBC	5	O	10-2/10-5
LAD[0:31]	—	LBC address/data	LBC	32	I/O	10-2/10-5
LALE	—	LBC address latch enable	LBC	1	O	10-2/10-5
LBCTL	—	LBC data buffer control	LBC	1	O	10-2/10-5
LCKE	—	LBC clock enable	LBC	1	O	10-2/10-5
LCLK[0]	—	LBC clocks 0	LBC	1	O	10-2/10-5
LCLK[1]/ LCS[6]	LCLK[1]	LBC clocks 1	LBC	1	O	10-2/10-5
	LCS[6]	LBC chip select 6				
LCLK[2]/ LCS[7]	LCLK[2]	LBC clocks 2	LBC	1	O	10-2/10-5
	LCS[7]	LBC chip select 7				
LCS[0:5]	—	LBC chip select 0–5	LBC	6	O	10-2/10-5
LDP[0]/ CKSTOP_OUT	LDP[0]	LBC data parity 0	LBC	1	I/O	10-2/10-5
	CKSTOP_OUT	Checkstop out	CPU		O	4-3/4-4
LDP[1]/ CKSTOP_IN	LDP[1]	LBC data parity 1	LBC	1	I/O	10-2/10-5
	CKSTOP_IN	Checkstop in	CPU		I	4-3/4-4
LDP[2]/LCS[6]	LDP[2]	LBC data parity 2	LBC	1	I/O	10-2/10-5
	LCS[6]	LBC chip select 6			O	10-2/10-5
LDP[3]/LCS[7]	LDP[3]	LBC data parity 3	LBC	1	I/O	10-2/10-5
	LCS[7]	LBC chip select 7			O	10-2/10-5
LGPL0/LSDA10/ cfg_reset_source0	LGPL0	LBC UPM general purpose line 0	LBC	1	O	10-2/10-5
	LSDA10	SDRAM address bit 10	LBC		O	10-2/10-5
	cfg_reset_source0	Configuration reset source 0	Reset & clock		I	4-5/4-9
LGPL1/LSDWE/ cfg_reset_source1	LGPL1	LBC UPM general purpose line 1	LBC	1	O	10-2/10-5
	LSDWE	SDRAM write enable	LBC		O	10-2/10-5
	cfg_reset_source1	Configuration reset source 1	Reset & clock		I	4-7/4-11

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
LGPL2/LSDRAS/LOE	LGPL2	LBC UPM general purpose line 2	LBC	1	O	10-2/10-5
	LSDRAS	SDRAM RAS			O	10-2/10-5
	LOE	LBC output enable			O	10-2/10-5
LGPL3/LSDCAS/cfg_reset_source2	LGPL3	LBC UPM general purpose line 3	LBC		O	10-2/10-5
	LSDCAS	SDRAM CAS	LBC		O	10-2/10-5
	cfg_reset_source2	Configuration reset source 2	Reset & clock		I	4-5/4-9
LGPL4/LGTA/ LUPWAIT/LPBSE	LGPL4	LBC UPM general purpose line 4	LBC	1	I/O	10-2/10-5
	LGTA	GPCM terminate access				
	LUPWAIT	UPM wait				
	LPBSE	LBC parity byte select				
LGPL5/ cfg_clkin_div	LGPL5	LBC UPM general purpose line 5	LBC	1	O	10-2/10-5
	cfg_clkin_div	Configuration clock in div	Reset & clock		I	4-7/4-11
LSYNC_IN	—	LBC DLL synchronization input	LBC	1	I	10-2/10-5
LSYNC_OUT	—	LBC DLL synchronization output	LBC	1	O	10-2/10-5
LWE[0:3]/ LSDDQM[0:3]/ LBS[0:3]	LWE[0:3]	LBC write enable 0–3	LBC	4	O	10-2/10-5
	LSDDQM[0:3]	Byte lane data mask 0–3			O	3-4/3-19
	LBS[0:3]	Byte select 0–3			O	10-2/10-5
M66EN/CE_PF[4]	M66EN	PCI 66-MHz timing on/off.	PCI	1	I	13-3/13-5
	CE_PF[4]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
MCP_OUT	—	Machine check interrupt output	IPIC	1	O	8-2/8-5
MEMC1_MA[0:14]	—	Primary DDR address	DDR	15	O	9-3/9-6
MEMC1_MBA[0:2]	—	Primary DDR bank select	DDR	3	O	9-3/9-6
MEMC1_MCAS	—	Primary DDR column address strobe	DDR	1	O	9-3/9-6
MEMC1_MCK[0:1]	—	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
MEMC1_MCK[0:1]	—	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
MEMC1_MCK[2:3]/ MEMC2_MCK[0:1]	MEMC1_MCK[2:3]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCK[0:1]	Secondary DDR differential clocks (positive)	DDR		O	9-4/9-9

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
$\overline{\text{MEMC1\_MCK}}$[2:3]/ $\overline{\text{MEMC2\_MCK}}$[0:1]	$\overline{\text{MEMC1\_MCK}}$[2:3]	Primary DDR differential clocks (negative)	DDR	2	O	9-4/9-9
	$\overline{\text{MEMC2\_MCK}}$[0:1]	Secondary DDR differential clocks (negative)	DDR		O	9-4/9-9
MEMC1_MCK[4:5]/ MEMC2_MCKE[0:1]	MEMC1_MCK[4:5]	Primary DDR differential clocks (positive)	DDR	2	O	9-4/9-9
	MEMC2_MCKE[0:1]	Secondary DDR clock enable			O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[4]/ MEMC2_MDM[8]	$\overline{\text{MEMC1\_MCK}}$[4]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDM[8]	Secondary DDR ECC data mask	DDR		O	9-4/9-9
$\overline{\text{MEMC1\_MCK}}$[5]/ MEMC2_MDQS[8]	$\overline{\text{MEMC1\_MCK}}$[5]	Primary DDR differential clocks (negative)	DDR	1	O	9-4/9-9
	MEMC2_MDQS[8]	Secondary DDR data strobe	DDR		I/O	9-4/9-9
MEMC1_MCKE[0:1]	—	Primary DDR clock enable	DDR	2	O	9-4/9-9
$\overline{\text{MEMC1\_MCS}}$[0:1]	—	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
MEMC1_MDIC[0:1]	—	DRAM driver impedance calibration	DDR	2	I/O	9-3/9-6
$\overline{\text{MEMC1\_MCS}}$[2:3]/ $\overline{\text{MEMC2\_MCS}}$[0:1]	$\overline{\text{MEMC1\_MCS}}$[2:3]	Primary DDR chip select (2/DIMM)	DDR	2	O	9-3/9-6
	$\overline{\text{MEMC2\_MCS}}$[0:1]	Secondary DDR chip select (2/DIMM)			O	9-3/9-6
MEMC1_MDM[0:3]	—	Primary DDR data mask 0–3	DDR	4	O	9-3/9-6
MEMC1_MDM[4:7]/ MEMC2_MDM[0:3]	MEMC1_MDM[4:7]	Primary DDR data mask 4–7	DDR	4	O	9-3/9-6
	MEMC2_MDM[0:3]	Secondary DDR data mask 0–3			O	9-3/9-6
MEMC1_MDM[8]	—	Primary DDR ECC data mask	DDR	1	O	9-3/9-6
MEMC1_MDQ[0:31]	—	Primary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQ[32:63]/ MEMC2_MDQ[0:31]	MEMC1_MDQ[32:63]	Primary DDR data	DDR	32	I/O	9-3/9-6
	MEMC2_MDQ[0:31]	Secondary DDR data	DDR	32	I/O	9-3/9-6
MEMC1_MDQS[0:3]	—	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
MEMC1_MDQS[4:7]/ MEMC2_MDQS[0:3]	MEMC1_MDQS[4:7]	Primary DDR data strobe	DDR	4	I/O	9-3/9-6
	MEMC2_MDQS[0:3]	Secondary DDR data strobe			I/O	9-3/9-6
MEMC1_MDQS[8]	—	Primary DDR data strobe	DDR	1	I/O	9-3/9-6
MEMC1_MECC[0:4]	MEMC1_MSRCID[0:4]	Primary DDR error correcting code	DDR	5	I/O	9-3/9-6

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
MEMC1_MECC[5]	MEMC1_MDVAL	Primary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC1_MECC[6:7]	—	Primary DDR error correcting code	DDR	3	I/O	9-3/9-6
MEMC1_MODT[0:3]	—	On-die termination for DRAM chip	DDR	4	O	9-3/9-6
$\overline{\text{MEMC1\_MRAS}}$	—	Primary DDR row address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC1\_MWE}}$	—	Primary DDR write enable	DDR	1	O	9-3/9-6
MEMC2_MA[0:14]	—	Secondary DDR address	DDR	15	O	9-3/9-6
MEMC2_MBA[0:2]	—	Secondary DDR bank select	DDR	3	O	9-3/9-6
$\overline{\text{MEMC2\_MCAS}}$	—	Secondary DDR column address strobe	DDR	1	O	9-3/9-6
MEMC2_MECC[0:4]	MEMC2_MSRCID[0:4]	Secondary DDR error correcting code	DDR	5	I/O	9-3/9-6
MEMC2_MECC[5]	MEMC2_MDVAL	Secondary DDR error correcting code	DDR	1	I/O	9-3/9-6
MEMC2_MECC[6:7]	—	Secondary DDR error correcting code	DDR	3	I/O	9-3/9-6
$\overline{\text{MEMC2\_MRAS}}$	—	Secondary DDR row address strobe	DDR	1	O	9-3/9-6
$\overline{\text{MEMC1\_MWE}}$	—	Primary DDR write enable	DDR	1	O	9-3/9-6
PCI_AD[31:0]/ CE_PG[31:0]	PCI_AD[31:0]	PCI address/data	PCI	32	I/O	13-3/13-5
	CE_PG[31:0]	QUICC Engine parallel I/O	PIO		I/O	3-18/3-61
PCI_C/$\overline{\text{BE}}$[3:0]/ CE_PF[10:7]	PCI_C	PCI command	PCI	4	I/O	13-3/13-5
	$\overline{\text{BE}}$[3:0]	Byte enable	PCI		I/O	13-3/13-5
	CE_PF[10:7]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_CLK_OUT[0:2] CE_PF[26:28]	PCI_CLK_OUT[0:2]	PCI clock out 0–2	Clocks	3	O	4-4/4-5
	CE_PF[26:28]	QUICC Engine port F	QUICC Engine		I/O	3-17/3-56
PCI_CLOCK/ PCI_SYNC_IN	PCI_CLOCK	PCI clock	Clocks	1	I	4-2/4-3
	PCI_SYNC_IN	PCI clock sync input	Clocks			
$\overline{\text{PCI\_DEVSEL}}$/ CE_PF[16]	$\overline{\text{PCI\_DEVSEL}}$	PCI device select	PCI	1	I/O	13-3/13-5
	CE_PF[16]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
$\overline{\text{PCI\_FRAME}}$/ CE_PF[12]	$\overline{\text{PCI\_FRAME}}$	PCI frame	PCI	1	I/O	13-3/13-5
	CE_PF[12]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_GNT[0]/ CE_PF[23]	PCI_GNT[0]	PCI grant 0	PCI	1	I/O	13-3/13-5
	CE_PF[23]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_GNT[1]/ CPCI_HS_LED/ CE_PF[24]	PCI_GNT[1]	PCI grant 1	PCI	1	O	13-3/13-5
	CPCI_HS_LED	Compact PCI hot swap LED	CPCI		O	13-3/13-5
	CE_PF[24]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_GNT[2]/ CPCI_HS_ENUM/ CE_PF[25]	PCI_GNT[2]	PCI grant 2	PCI	1	O	13-3/13-5
	CPCI_HS_ENUM	Compact PCI hot swap enumerator	CPCI		O	13-3/13-5
	CE_PF[25]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_IDSEL/ CE_PF[17]	PCI_IDSEL	PCI initial device select	PCI	1	I	13-3/13-5
	CE_PF[17]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_INTA/IRQ_OUT/ CE_PF[5]	PCI_INTA	PCI interrupt output	PCI	1	O	13-3/13-5
	IRQ_OUT	Interrupt output	IPIC		O	13-3/13-5
	CE_PF[5]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_IRDY/ CE_PF[14]	PCI_IRDY	PCI initiator ready	PCI	1	I/O	13-3/13-5
	CE_PF[14]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_MODE	—	PCI mode select	PCI	1	I	5-32/5-24
PCI_PAR/CE_PF[11]	PCI_PAR	PCI parity	PCI	1	I/O	13-3/13-5
	CE_PF[11]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_PERR/ CE_PF[19]	PCI_PERR	PCI parity error	PCI	1	I/O	13-3/13-5
	CE_PF[19]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_REQ[0]/ CE_PF[20]	PCI_REQ[0]	PCI request 0	PCI	1	I/O	13-3/13-5
	CE_PF[20]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_REQ[1]/ CPCI_HS_ES/ CE_PF[21]	PCI_REQ[1]	PCI request 1	PCI	1	I	13-3/13-5
	CPCI_HS_ES	Compact PCI hot swap ejector switch	CPCI		I	13-3/13-5
	CE_PF[21]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_REQ[2]/ CE_PF[22]	PCI_REQ[2]	PCI request 2	PCI	1	I	13-3/13-5
	CE_PF[22]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_RESET_OUT/ CE_PF[6]	PCI_RESET_OUT	PCI reset	PCI	1	O	13-3/13-5
	CE_PF[6]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_SERR/ CE_PF[18]	PCI_SERR	PCI system error	PCI	1	I/O	13-3/13-5
	CE_PF[18]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56

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

Signal Name	Alternate Function(s)	Description	Functional Block	No. of Signals	I/O	Table/Page
PCI_STOP/ CE_PF[15]	PCI_STOP	PCI stop	PCI	1	I/O	13-3/13-5
	CE_PF[15]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_SYNC_IN/ PCI_CLOCK	PCI_SYNC_IN	PCI clock sync input	Clocks	1	I	4-4/4-5
	PCI_CLOCK	PCI clock	Clocks			
PCI_SYNC_OUT/ CE_PF[29]	PCI_SYNC_OUT	PCI clock sync output	Clocks	1	O	4-4/4-5
	CE_PF[29]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PCI_TRDY/ CE_PF[13]	PCI_TRDY	PCI target ready	PCI	1	I/O	13-3/13-5
	CE_PF[13]	QUICC Engine parallel I/O	PIO		I/O	3-17/3-56
PORESET	—	Power-on reset	System control	1	I	4-3/4-4
QUIESCE	—	Quiesce state	PMC	1	O	5-77/5-73
SRESET	—	Soft reset	System control	1	I/O	4-3/4-4
TCK	—	Test clock	JTAG	1	I	17-2/17-2
TDI	—	Test data in	JTAG	1	I	17-2/17-2
TDO	—	Test data out	JTAG	1	O	17-2/17-2
TEST_SEL (MPC8360) TEST_SEL (MPC8358)	—	Test mode	TEST	1	I	—
TEST	—	Test mode	TEST	1	I	—
TMS	—	Test mode select	JTAG	1	I	17-2/17-2
TRST	—	Test reset	JTAG	1	I	17-2/17-2
UART1_CTS/ M1SRCID[2]/ M2SRCID[2] LSRCID[2]	UART1_CTS	UART1 clear to send	UART	1	I	16-2/16-3
	M1SRCID2	DDR memory debug source port ID 2	DDR		O	9-7/9-13
	M2SRCID2	Secondary DDR memory debug source port ID 2	DDR		O	9-7/9-13
	LSRCID2	LBC debug source port ID 2	LBC		O	10-2/10-5
UART1_RTS M1SRCID[3]/ M2SRCID[3] LSRCID[3]	UART1_RTS	UART1 ready to send	UART	1	O	16-2/16-3
	M1SRCID3	DDR memory debug source port ID 3	DDR		O	9-7/9-13
	M2SRCID3	Secondary DDR memory debug source port ID 3	DDR		O	9-7/9-13
	LSRCID3	LBC debug source port ID 3	LBC		O	10-2/10-5

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

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

3.2 Configuration Signals Sampled at Reset

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

Table 3-3. MPC8360E Reset Configuration Signals

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

3.3 Output Signal States During Reset

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

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset
MEMCx_MDM[0:8]	DDR data mask	All ‘0’
MEMCx_MBA[0:1]	DDR bank select	All ‘Z’
MEMCx_MA[0:14]	DDR address	All ‘Z’
$\overline{\text{MEMCx\_MWE}}$	DDR write enable	‘Z’
$\overline{\text{MEMCx\_MRAS}}$	DDR row address strobe	‘Z
$\overline{\text{MEMCx\_MCAS}}$	DDR column address strobe	‘Z’
$\overline{\text{MEMCx\_MCS}}$[0:3]	DDR chip select (2/DIMM)	All ‘Z’
MEMCx_MCKE[0:1]	DDR clock enable	All ‘0’
MEMCx_MCK[0:5]	DDR differential clocks	All ‘0’
$\overline{\text{MEMCx\_MCK}}$[0:5]	DDR differential clocks	All ‘0’
MEMC1_MODT[0:3]	DRAM on-die termination	All ‘0’
$\overline{\text{INTA}}$/ $\overline{\text{IRQ\_OUT}}$	PCI interrupt output	High impedance
$\overline{\text{PCI\_RESET\_OUT}}$	PCI reset output	‘0’
$\overline{\text{PCI\_GNT}}$[0:2]	PCI grant 0–2	High impedance
UART1_SOUT	UART1 serial data out	‘1’
$\overline{\text{UART1\_RTS}}$	UART1 ready to send	‘1’
LA[27:31]	LBC port address	Active – being used to load reset configuration word
$\overline{\text{LCS}}$[0]	LBC chip select 0	Active – being used to load reset configuration word
$\overline{\text{LCS}}$[1:7]	LBC chip select 1–7	‘1111111’
$\overline{\text{LWE}}$[0:3]/ LSDDQM[0:3]/ $\overline{\text{LBS}}$[0:3]	LBC write enable/byte lane data mask/byte select	‘1111’
LBCTL	LBC data buffer control	Active – being used to load reset configuration word
LALE	LBC address latch enable	Active – being used to load reset configuration word
LGPL2/$\overline{\text{LSDRAS}}$/ $\overline{\text{LOE}}$	LBC output enable/SDRAM RAS/GP line 2	Active – being used to load reset configuration word
LCKE	LBC clock enable	‘1’
LCLK[0:2]	LBC clock	‘0’
LSYNC_OUT	LBC clock synchronization output	‘0’
$\overline{\text{MCP\_OUT}}$	Machine check interrupt output	High impedance

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

Interface	Signal	State During Reset
TDO	Test data out	High impedance
$\overline{\text{QUIESCE}}$	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
PCI (CE_PF[4:29] & CE_PG[0:31])	Signal $\overline{\text{PCI\_MODE}}$	5-32/5-24
PCI clock outputs (CE_PF[26:28])	RCWH[PCICKDRV] (In addition to $\overline{\text{PCI\_MODE}}$ as described above)	4-21/4-21
PCI/CPCI	RCWH[PCIARB] (In addition to $\overline{\text{PCI\_MODE}}$ as described above)	4-12/4-16
QUICC Engine block	CPPARxA–CPPARxG	3-9/3-25
All others	SICRL/SICRH	5-40/5-29, 5-41/5-31

3.4 Parallel I/O Ports

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

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

3.4.1 Features

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

- Ports A, E, and G have 32 pins.
- Ports C and F have 30 pins.
- Ports B and D have 28 pins.
- All ports are bidirectional.

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

3.4.2 Port Registers

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

The port registers are: 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.

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

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

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

Offset	0x1400 (CPODRA), 0x1418 (CPODRB), 0x1430 (CPODRC), 0x1448 (CPODRD), 0x1460 (CPODRE), 0x1478 (CPODRF), 0x1490 (CPODRG)															
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–CPODRG)

Table 3-6. CPODRx Field Descriptions

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

3.4.2.2 QUICC Engine Port Data Registers (CPDATA–CPDATG)

A read of a QUICC Engine port data register (CPDAT*x*), shown in Figure 3-3, 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), 0x1464 (CPDATE), 0x147C (CPDATF), 0x1494 (CPDATG)															
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–CPDATG)

Table 3-7. CPDATx Field Descriptions

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

3.4.2.3 QUICC Engine Port Direction Registers (CPDIRxA–CPDIRxG)

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

offset	0x1408 (CPDIR1A), 0x1420 (CPDIR1B), 0x1438 (CPDIR1C), 0x1450 (CPDIR1D), 0x1468 (CPDIR1E), 0x1480 (CPDIR1F), 0x1498 (CPDIR1G)															
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–CPDIR1G)

offset	0x140C (CPDIR2A), 0x1424 (CPDIR2B), 0x143C (CPDIR2C), 0x1454 (CPDIR2D), 0x146C (CPDIR2E), 0x1484 (CPDIR2F), 0x149C (CPDIR2G)															
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–CPDIR2G)

Table 3-8. CPDIRxy Field Descriptions

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

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

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

offset	0x1410 (CPPAR1A), 0x1428 (CPPAR1B), 0x1440 (CPPAR1C), 0x1458 (CPPAR1D), 0x1470 (CPPAR1E), 0x1488 (CPPAR1F), 0x14A0 (CPPAR1G)															
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–CPPAR1G)

offset	0x1414 (CPPAR2A), 0x142C (CPPAR2B), 0x1444 (CPPAR2C), 0x145C (CPPAR2D), 0x1474 (CPPAR2E), 0x148C (CPPAR2F), 0x14A4 (CPPAR2G)															
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–CPPAR2G)

Table 3-9. CPPARxy Field Descriptions

Bits	Name	Description
0–1, 2–3, etc.	SEL*n*	Determines the function of pin number *n*, according to Table 3-12 through Table 3-18 below. Functions not shown in the table represent Reserved values of these bits.

3.4.3 Port Block Diagram

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

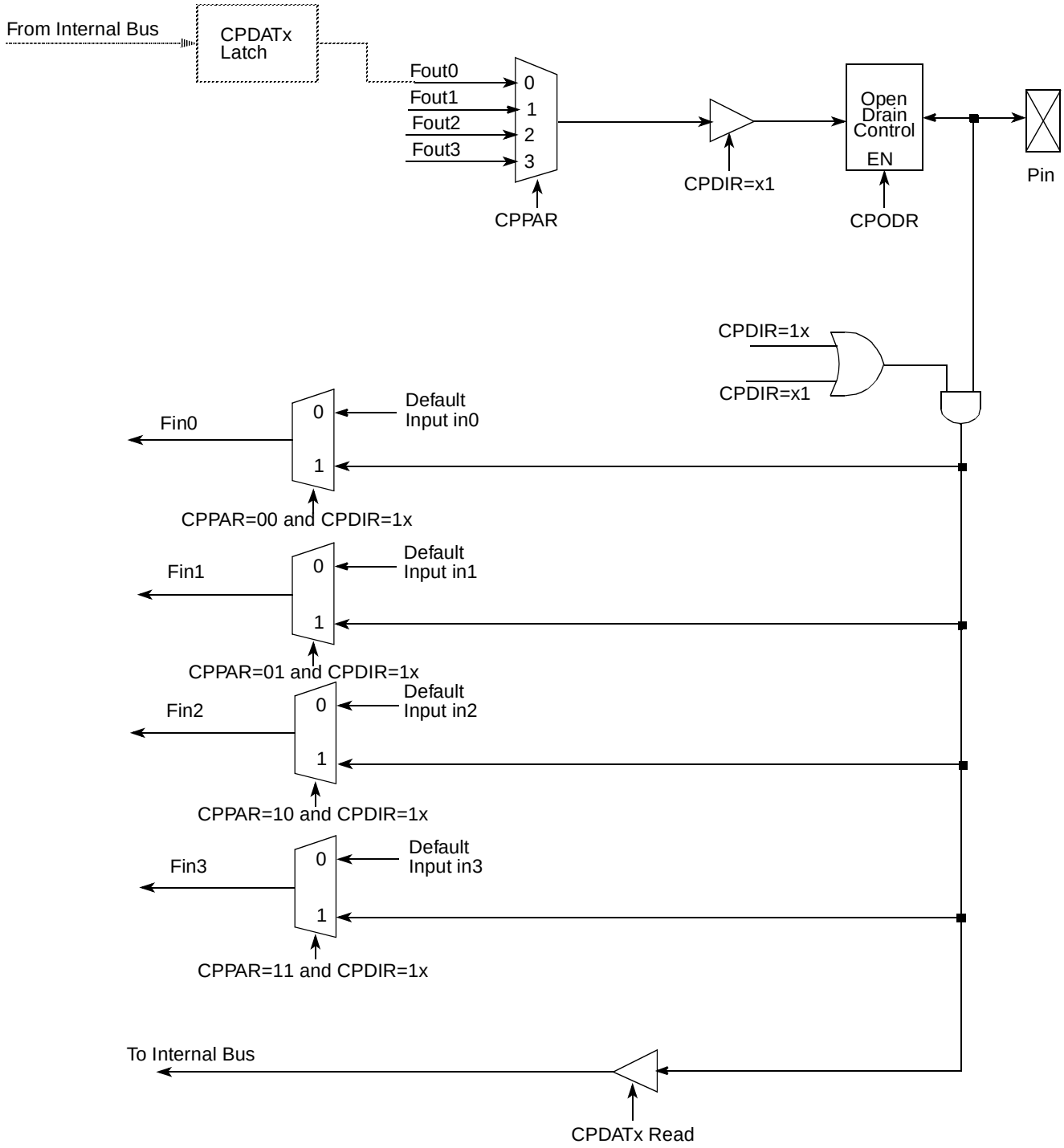


Figure 3-8. Port Functional Block Diagram

3.4.4 Port Pins Functions

Generally each pin can function as a general purpose I/O pin or as a dedicated input or output pin. Note however that some pins have only “General Purpose Input” or “General Purpose Output” function (not both possibilities). Refer to Table 3-12 through Table 3-18 for details. Usually following hard reset all port pins are disabled—both output and input buffers are off.

3.4.4.1 General-Purpose I/O Pins

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

3.4.4.2 Dedicated Pins

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

NOTE

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

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

3.4.5 PCI Pins

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

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

3.4.6 QUICC Engine Ports Interrupts

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

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

3.4.7 Gigabit Ethernet Pins

3.4.7.1 RGMII pins

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

Table 3-10. RGMII Pins

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

3.4.7.2 GMII and TBI pins

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

Table 3-11. GMII and TBI Pins

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

3.4.8 Ports Tables

Table 3-12 through Table 3-18 describe the ports functionality according to the configuration of the port registers (CPPARxy and CPDIRxy).

Some input functions can come from two or three different pins for flexibility. Figure 3-9 shows an example in which three different pins can be selected for one input function. Secondary option programming is relevant only if primary option is programmed to the default value. Third option programming is relevant only if both primary and secondary options are programmed to the default value.

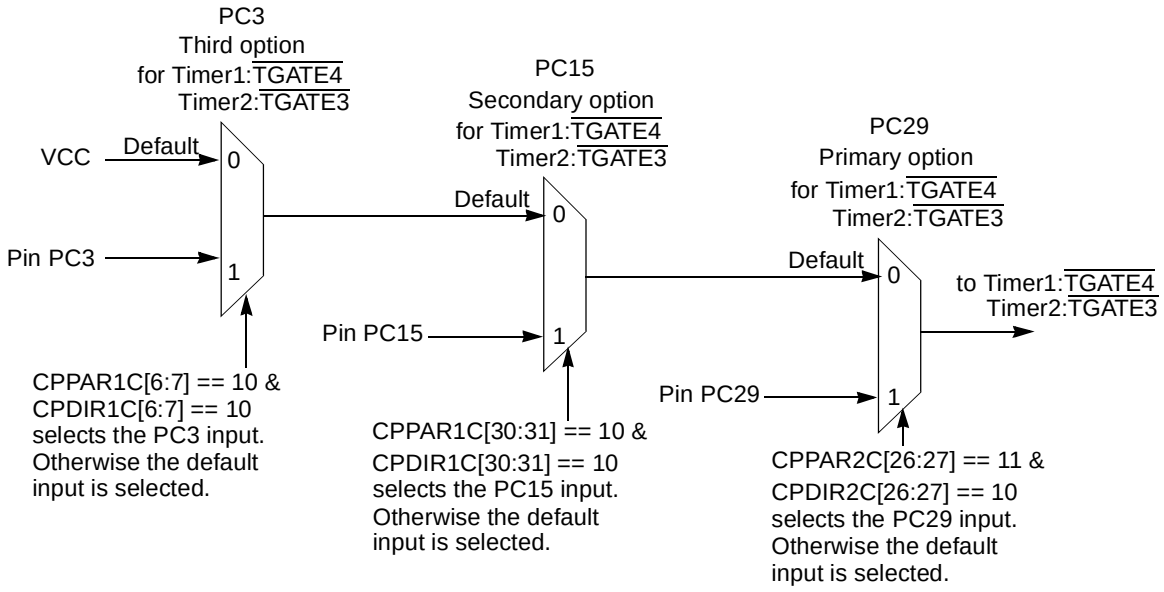


Figure 3-9. Primary, Secondary and Third Option Programming

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

Table 3-12. Port A Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARAx[SELn]=00			CPPARAx[SELn]=01			CPPARAx[SELn]=10			CPPARAx[SELn]=11		
		Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input	Function	CPDIRxA[DIRn]	Default Input
PA0	IN	GPI_PA0	10		UCC1:Gigabit RX_CLK Enet	10	GND						
	OUT	GPO_PA0	01										
PA1	IN	GPI_PA1	10		SPI2:MDIO	11	GND	CE MUX:MDIO	11	GND			
	OUT	GPO_PA1	01										
PA2	IN	GPI_PA2	10										
	OUT	GPO_PA2	01		CE MUX:MDC	01		SPI2:MDC	01				
PA3	IN	GPI_PA3	10										
	OUT	GPO_PA3	01		UCC1:TxD[0]/TCG[0] Enet UCC1:TxD[0]/ UCC1:TxD (Serial) SER	01		TDMa:TxD[1]	01				

Table 3-12. 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
PA4	IN	GPI_PA4	10										
	OUT	GPO_PA4	01		UCC1:TxD[1]/TCG[1] Enet UCC1:TxD[1] SER	01		TDMa:TxD[2]	01				
PA5	IN	GPI_PA5	10										
	OUT	GPO_PA5	01		UCC1:TxD[2]/TCG[2] Enet UCC1:TxD[2] SER	01		TDMa:$\overline{RQ}$	01				
PA6	IN	GPI_PA6	10					TDMa:TSYNC/GRANT (Secondary Option)	10	GND			
	OUT	GPO_PA6	01		UCC1:TxD[3]/TCG[3] Enet UCC1:TxD[3] SER	01							
PA7	IN	GPI_PA7	10										
	OUT	GPO_PA7	01		UCC1:TX_EN/TCG[8] Enet UCC1:$\overline{RTS}$ SER	01							
PA8	IN	GPI_PA8	10		UPC1:RxClav[1] UTOPIA UPC1:PRPA[1]/ DRPA[1] POS (Secondary Option)	10	GND	TDMa:RxD[0]/ TDMa:RxD (Serial) (Secondary Option)	11	GND	UPC2:TxAddr[1] UTOPIA slave UPC2:TADR[1] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA8	01		UCC1:TX_ER/TCG[9] Enet	01					UPC2:RxAddr[1] UTOPIA master UPC2:RADR[1] POS master	01	
PA9	IN	GPI_PA9	10		UCC1:RxD[0]/RCG[0] Enet UCC1:RxD[0]/ UCC1:RxD (Serial) SER	10	GND	TDMa:RxD[1]	10	GND			
	OUT	GPO_PA9	01										
PA10	IN	GPI_PA10	10		UCC1:RxD[1]/RCG[1] Enet UCC1:RxD[1] SER	10	GND	TDMa:RxD[2]	10	GND			
	OUT	GPO_PA10	01										
PA11	IN	GPI_PA11	10		UCC1:RxD[2]/RCG[2] Enet UCC1:RxD[2] SER	10	GND						
	OUT	GPO_PA11	01					SI:STRB1	01				

Table 3-12. 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
PA12	IN	GPI_PA12	10		UCC1:RxD[3]/RCG[3] Enet UCC1:RxD[3] SER	10	GND						
	OUT	GPO_PA12	01					TDMa:CLKO	01				
PA13	IN	GPI_PA13	10		UCC1:RxD[4]/RCG[4] COL Enet UCC1:RxD[4] SER	10	GND	TDMa:TxD[0]/ TDMa:TxD (Serial) (Secondary Option)	11	GND	UPC1:TxClav[1] UTOPIA UPC1:PTPA[1]/DTPA[1] POS (Secondary Option)	10	GND
	OUT	GPO_PA13	01								UPC2:TxData[4] UTOPIA 16[1] UPC2:TDAT[4] POS	01	
PA14	IN	GPI_PA14	10		UCC1:CRS/SDET Enet	10	GND	TDMa:RSYNC (Secondary Option)	10	GND	UPC2:TxAddr[3] UTOPIA slave UPC2:TADR[3] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA14	01		UPC1:$\overline{\text{RxEnb}}$[1] UTOPIA UPC1:RENB[1] POS	01					UPC2:RxAddr[3] UTOPIA master UPC2:RADR[3] POS master	01	
PA15	IN	GPI_PA15	10		UCC1:RX_DV/RCG[8] Enet UCC1:$\overline{\text{CTS}}$ SER	10	GND	TDMa:RxD[3]	10	GND			
	OUT	GPO_PA15	01										
PA16	IN	GPI_PA16	10		UCC1:RX_ER/RCG[9] Enet UCC1:$\overline{\text{CD}}$ SER	10	GND	TDMa:TSYNC/GRANT (Primary Option)	10	PA6	UPC2:RxAddr[3] UTOPIA slave UPC2:RADR[3] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA16	01		UPC1:$\overline{\text{TxEnb}}$[1] UTOPIA UPC1:TENB[1] POS	01		TDMa:TxD[3]	01		UPC2:TxAddr[3] UTOPIA master UPC2:TADR[3] POS master	01	
PA17	IN	GPI_PA17	10								TDMe:TSYNC/GRANT (Third Option)	10	GND
	OUT	GPO_PA17	01		UCC2:TxD[0]/TCG[0] Enet UCC2:TxD[0]/ UCC2:TxD (Serial) SER	01					TDMb:TxD[1]	01	

Table 3-12. 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
PA18	IN	GPI_PA18	10								TDMd:TSYNC/GRANT (Third Option)	10	GND
	OUT	GPO_PA18	01		UCC2:TxD[1]/TCG[1] Enet UCC2:TxD[1] SER	01		TDMb:TxD[2]	01				
PA19	IN	GPI_PA19	10					UPC2:RERR POS master UPC2:TERR POS slave (Secondary Option)	10	GND	TDMd:TxD[0]/ TDMd:TxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA19	01		UCC2:TxD[2]/TCG[2] Enet UCC2:TxD[2] SER	01		TDMb:$\overline{RQ}$	01				
PA20	IN	GPI_PA20	10		UPC2:STPA POS master (Secondary Option)	10	GND	TDMb:TSYNC/GRANT (Secondary Option)	10	GND			
	OUT	GPO_PA20	01		UCC2:TxD[3]/TCG[3] Enet UCC2:TxD[3] SER	01		UPC2:STPA POS slave	01				
PA21	IN	GPI_PA21	10					TDMf:TSYNC/GRANT (Third Option)	10	GND			
	OUT	GPO_PA21	01		UCC2:TX_EN/TCG[8] Enet UCC2:$\overline{RTS}$ SER	01							
PA22	IN	GPI_PA22	10		UPC1:RxClav[2] UTOPIA UPC1:PRPA[2]/ DRPA[2] POS (Secondary Option)	10	GND	TDMb:TxD[0] TDMb:TxD (Secondary Option)	11	GND	UPC2:TxAddr[2] UTOPIA slave UPC2:TADR[2] POS slave (Secondary Option)	10	GND
	OUT	GPO_PA22	01		UCC2:TX_ER/TCG[9] Enet	01					UPC2:RxAddr[2] UTOPIA master UPC2:RADR[2] POS master	01	
PA23	IN	GPI_PA23	10		UCC2:RxD[0]/RCG[0] Enet UCC2:RxD[0]/ UCC2:RxD (Serial) SER (Primary Option)	10	PF23	TDMb:RxD[1]	10	GND	TDMf:RSYNC (Third Option)	10	GND
	OUT	GPO_PA23	01										

Table 3-12. 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
PA24	IN	GPI_PA24	10		UCC2:RxD[1]/RCG[1] Enet UCC2:RxD[1] SER (Primary Option)	10	PF24	TDMb:RxD[2]	10	GND	TDMd:RSYNC (Third Option)	10	GND
	OUT	GPO_PA24	01										
PA25	IN	GPI_PA25	10		UCC2:RxD[2]/RCG[2] Enet UCC2:RxD[2] SER (Primary Option)	10	PG30	TDMd:RxD[0]/ TDMd:RxD (Serial) (Third Option)	11	GND			
	OUT	GPO_PA25	01		SI:STRB2	01							
PA26	IN	GPI_PA26	10		UCC2:RxD[3]/RCG[3] Enet UCC2:RxD[3] SER (Primary Option)	10	PF25	TDMe:RSYNC (Third Option)	10	GND			
	OUT	GPO_PA26	01		UPC2:TMOD POS master UPC2:RMOD POS slave	01		TDMb:CLKO	01				
PA27	IN	GPI_PA27	10		UCC2:RxD[4]/RCG[4] COL Enet UCC2:RxD[4] SER (Primary Option)	10	PG3	TDMb:RxD[0]/ TDMb:RxD (Serial) (Secondary Option)	11	GND	UPC1:TxClav[2] UTOPIA UPC1:PTPA[2]/DTPA[2] POS (Secondary Option)	10	GND
	OUT	GPO_PA27	01		UPC2:TxSOC UTOPIA master UPC2:RxSOC UTOPIA slave UPC2:TSOP POS master UPC2:RSOP POS slave	01							
PA28	IN	GPI_PA28	10		UCC2:CRS/SDET Enet (Primary Option)	10	PF28	TDMb:RSYNC (Secondary Option)	10	GND			
	OUT	GPO_PA28	01		$\overline{\text{DMA\_DDONE0}}$ (Secondary Option)	01		UPC2:RxAddr[5] UTOPIA UPC2:RADR[5] POS	01		Timer1:$\overline{\text{TOUT1}}$	01	
PA29	IN	GPI_PA29	10		UCC2:RX_DV/RCG[8] Enet UCC2:$\overline{\text{CTS}}$ SER (Primary Option)	10	PG31	TDMb:RxD[3]	10	GND	TDMe:RxD[0]/ TDMe:RxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA29	01										

Table 3-12. 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
PA30	IN	GPI_PA30	10		UCC2:RX_ER/RCG[9] Enet UCC2:$\overline{CD}$ SER (Primary Option)	10	PF20	TDMb:TSYNC/GRANT (Primary Option)	10	PA20	TDMe:TxD[0]/ TDMe:TxD (Serial) (Third Option)	11	GND
	OUT	GPO_PA30	01		$\overline{DMA\_DDONE3}$ (Secondary Option)	01		TDMb:TxD[3]	01				
PA31	IN	GPI_PA31	10		UCC2:Gigabit RX_CLK Enet (Secondary Option)	10	GND	UPC2:RVAL POS master (Secondary Option)	10	GND	Timer1:$\overline{TGATE2}$ Timer2:$\overline{TGATE1}$ (Secondary Option)	10	VCC
	OUT	GPO_PA31	01					UPC2:RVAL POS slave	01				

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

Table 3-13. Port B Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB0	IN	GPI_PB0	10		TDMd:RSYNC (Secondary Option)	10	PA24	UCC1:RxD[6]/RCG[6] Enet UCC1:RxD[6] SER (Secondary Option)	10	GND	UPC2:TxAddr[0] UTOPIA slave UPC2:TADR[0] POS slave (Secondary Option)	10	GND
	OUT	GPO_PB0	01		UCC3:TxD[0] Enet UCC3:TxD[0]/ UCC3:TxD (Serial) SER	01		TDMc:TxD[1]	01		UPC2:RxAddr[0] UTOPIA master UPC2:RADR[0] POS master	01	
PB1	IN	GPI_PB1	10		UPC2:RxData[1] UTOPIA 8[1] UPC2:RxData[9] UTOPIA 16[1] UPC2:RDAT[9] POS (Secondary Option)	10	GND	UCC1:RxD[5]/RCG[5] Enet UCC1:RxD[5] SER (Secondary Option)	10	GND	TDMd:TxD[0]/ TDMd:TxD (Serial) (Secondary Option)	11	PA19
	OUT	GPO_PB1	01		UCC3:TxD[1] Enet UCC3:TxD[1] SER	01		TDMc:TxD[2]	01				

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

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB2	IN	GPI_PB2	10		UCC4:RxD[1] Enet UCC4:RxD[1] SER (Secondary Option)	10	GND	UPC2:RxData[7] UTOPIA 8[1] UPC2:RxData[15] UTOPIA 16[1] UPC2:RDAT[15] POS (Secondary Option)	10	GND	TDMc:RSYNC (Secondary Option)	10	GND
	OUT	GPO_PB2	01		UCC2:TxD[4]/TCG[4] Enet UCC2:TxD[4] SER	01		UCC3:TxD[2] Enet UCC3:TxD[2] SER	01		USB:$\overline{OE}$	01	
PB3	IN	GPI_PB3	10		UPC2:RxData[6] UTOPIA 8[1] UPC2:RxData[14] UTOPIA 16[1] UPC2:RDAT[14][1] POS (Secondary Option)	10	GND	TDMc:TSYNC/GRANT (Secondary Option)	10	GND	UCC4:RX_DV Enet UCC4:$\overline{CTS}$ SER (Secondary Option)	10	GND
	OUT	GPO_PB3	01		UCC3:TxD[3] Enet UCC3:TxD[3] SER	01		UCC2:TxD[5]/TCG[5] Enet UCC2:TxD[5] SER	01		USB:TP	01	
PB4	IN	GPI_PB4	10		Timer1:TIN3 Timer2:TIN4 1588_EXT_TRIG1 (Secondary Option)	10	GND	UCC1:RxD[7]/RCG[7] Enet UCC1:RxD[7] SER (Secondary Option)	10	GND	UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Secondary Option)	10	GND
	OUT	GPO_PB4	01		UCC3:TX_EN Enet UCC3:$\overline{RTS}$ SER	01		$\overline{DMA\_DACK2}$ (Secondary Option)	01		UPC2:TxAddr[0] UTOPIA master UPC2:TADR[0] POS master	01	
PB5	IN	GPI_PB5	10		$\overline{DMA\_DREQ2}$ (Secondary Option)	10	VCC	TDMc:RxD[0]/ TDMc:RxD (Serial) (Secondary Option)	11	GND	UCC4:RX_ER Enet UCC4:$\overline{CD}$ SER (Secondary Option)	10	GND
	OUT	GPO_PB5	01		UCC3:TX_ER Enet	01					UCC2:TxD[6]/TCG[6] Enet UCC2:TxD[6] SER	01	
PB6	IN	GPI_PB6	10		UCC3:RxD[0] Enet UCC3:RxD[0]/ UCC3:RxD (Serial) SER	10	GND	TDMc:RxD[1]	10	GND	TDMd:TSYNC/GRANT (Secondary Option)	10	PA18
	OUT	GPO_PB6	01					$\overline{DMA\_DDONE2}$ (Secondary Option)	01		UCC1:TxD[4]/TCG[4] Enet UCC1:TxD[4] SER	01	

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

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB7	IN	GPI_PB7	10		UCC3:RxD[1] Enet UCC3:RxD[1] SER	10	GND	TDMc:RxD[2]	10	GND	TDMd:RxD[0]/ TDMd:RxD (Serial) (Secondary Option)	11	PA25
	OUT	GPO_PB7	01		UCC1:TxD[5]/TCG[5] Enet UCC1:TxD[5] SER	01		UPC2:TxData[6] UTOPIA 8[1] UPC2:TxData[14] UTOPIA 16[1] UPC2:TDAT[14] POS	01				
PB8	IN	GPI_PB8	10		UCC3:RxD[2] Enet UCC3:RxD[2] SER	10	GND	TDMc:TxD[0]/ TDMc:TxD (Serial) (Secondary Option)	11	GND	UCC4:RxD[0] Enet UCC4:RxD[0]/ UCC4:RxD (Serial) SER (Secondary Option)	10	GND
	OUT	GPO_PB8	01		USB:TN	01					UCC2:TxD[7]/TCG[7] Enet UCC2:TxD[7] SER	01	
PB9	IN	GPI_PB9	10		UCC3:RxD[3] Enet UCC3:RxD[3] SER	10	GND	UPC2:RxData[4] UTOPIA 8[1] UPC2:RxData[12] UTOPIA 16[1] UPC2:RDAT[12[1] POS (Secondary Option)	10	GND	USB:RP (Secondary Option)	10	GND
	OUT	GPO_PB9	01		TDMc:CLKO	01		UCC1:TxD[6]/TCG[6] Enet UCC1:TxD[6] SER	01		UCC4:TxD[0] Enet UCC4:TxD[0]/ UCC4:TxD (Serial) SER	01	
PB10	IN	GPI_PB10	10		UCC3:COL Enet	10	GND	UPC2:RxData[3] UTOPIA 8[1] UPC2:RxData[11] UTOPIA 16[1] UPC2:RDAT[11[1] POS (Secondary Option)	10	GND	USB:RXD (Secondary Option)	10	GND
	OUT	GPO_PB10	01		SI:STRB3	01		UCC1:TxD[7]/TCG[7] Enet UCC1:TxD[7] SER	01		UCC4:TxD[1] Enet UCC4:TxD[1] SER	01	

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

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB11	IN	GPI_PB11	10		UCC3:CRS Enet	10	GND	UCC2:RxD[7]/RCG[7] Enet UCC2:RxD[7] SER (Primary Option)	10	PF27	USB:RN (Secondary Option)	10	GND
	OUT	GPO_PB11	01		UPC2:TxData[2] UTOPIA 8 [1] UPC2:TxData[10] UTOPIA 16[1] UPC2:TDAT[10] POS	01		TDMc:$\overline{RQ}$	01		UCC4:TX_EN Enet UCC4:$\overline{RTS}$ SER	01	
PB12	IN	GPI_PB12	10		UCC3:RX_DV Enet UCC3:$\overline{CTS}$ SER	10	GND	UCC2:RxD[5]/RCG[5] Enet UCC2:RxD[5] SER (Primary Option)	10	PG13	UPC2:RxPrty UTOPIA master UPC2:TxPrty UTOPIA slave UPC2:RPRTY POS master UPC2:TPRTY POS slave (Secondary Option)	10	GND
	OUT	GPO_PB12	01		$\overline{DMA\_DACK3}$ (Secondary Option)	01		TDMc:TxD[3]	01				
PB13	IN	GPI_PB13	10		UCC3:RX_ER Enet UCC3:$\overline{CD}$ SER	10	GND	TDMc:RxD[3]	10	GND	UCC2:RxD[6]/RCG[6] Enet UCC2:RxD[6] SER (Primary Option)	10	PF29
	OUT	GPO_PB13	01					$\overline{DMA\_DDONE0}$ (Primary Option)	01		UPC2:TxPrty UTOPIA master UPC2:RxPrty UTOPIA slave UPC2:TPRTY POS master UPC2:RPRTY POS slave	01	
PB14	IN	GPI_PB14	10										
	OUT	GPO_PB14	01		UCC4:TxD[0] Enet UCC4:TxD[0]/ UCC4:TxD (Serial) SER	01		TDMd:TxD[1]	01		UPC1:TxSOC UTOPIA master UPC1:RxSOC UTOPIA slave UPC1:TSOP POS master UPC1:RSOP POS slave	01	
PB15	IN	GPI_PB15	10								UPC1:$\overline{RxEnb}$ UTOPIA slave UPC1:RENB POS slave	10	VCC
	OUT	GPO_PB15	01		UCC4:TxD[1] Enet UCC4:TxD[1] SER	01		TDMd:TxD[2]	01		UPC1:$\overline{TxEnb}$[0] UTOPIA master UPC1:TENB[0] POS master	01	

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

Pin	Pin Functions												
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB16	IN	GPI_PB16	10		UPC1:TxClav[0] UTOPIA master UPC1:PTPA[0]/DTPA[0] POS master	10	GND	TDMd:RSYNC (Primary Option)	10	PB0			
	OUT	GPO_PB16	01		UCC4:TxD[2] Enet UCC4:TxD[2] SER	01		UPC1:RxClav UTOPIA slave UPC1:PRPA/DRPA POS slave	01				
PB17	IN	GPI_PB17	10					TDMd:TSYNC/GRANT (Primary Option)	10	PB6			
	OUT	GPO_PB17	01		UCC4:TxD[3] Enet UCC4:TxD[3] SER	01		UPC1:TxData[7] UTOPIA 8[1] UPC1:TxData[15] UTOPIA 16[1] UPC1:TDAT[15] POS	01				
PB18	IN	GPI_PB18	10										
	OUT	GPO_PB18	01		UCC4:TX_EN Enet UCC4:$\overline{\text{RTS}}$ SER	01		UPC1:TxData[6] UTOPIA 8[1] UPC1:TxData[14] UTOPIA 16[1] UPC1:TDAT[14] POS	01				
PB19	IN	GPI_PB19	10		TDMd:TxD[0]/ TDMd:TxD (Serial) (Primary Option)	11	PB1						
	OUT	GPO_PB19	01					UCC4:TX_ER Enet	01		UPC1:TxData[5] UTOPIA 8[1] UPC1:TxData[13] UTOPIA 16[1] UPC1:TDAT[13] POS	01	
PB20	IN	GPI_PB20	10		UCC4:RxD[0] Enet UCC4:RxD[0]/ UCC4:RxD (Serial) SER (Primary Option)	10	PB8	TDMd:RxD[1]	10	GND			
	OUT	GPO_PB20	01					UPC1:TxData[4] UTOPIA 8[1] UPC1:TxData[12] UTOPIA 16[1] UPC1:TDAT[12] POS	01				

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

Pin	Pin Functions											
	Direction	CPPARBx[SELn]=00			CPPARBx[SELn]=01			CPPARBx[SELn]=10			CPPARBx[SELn]=11		
		Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input	Function	CPDIRxB[DIRn]	Default Input
PB21	IN	GPI_PB21	10		UCC4:RxD[1] Enet UCC4:RxD[1] SER (Primary Option)	10	PB2	TDMd:RxD[2]	10	GND			
	OUT	GPO_PB21	01					UPC1:TxData[3] UTOPIA 8[1] UPC1:TxData[11] UTOPIA 16[1] UPC1:TDAT[11] POS	01				
PB22	IN	GPI_PB22	10		UCC4:RxD[2] Enet UCC4:RxD[2] SER	10	GND	TDMd:RxD[0]/ TDMd:RxD (Serial) (Primary Option)	11	PB7			
	OUT	GPO_PB22	01		UPC1:TxData[2] UTOPIA 8 [1] UPC1:TxData[10] UTOPIA 16[1] UPC1:TDAT[10] POS	01							
PB23	IN	GPI_PB23	10		UCC4:RxD[3] Enet UCC4:RxD[3] SER	10	GND						
	OUT	GPO_PB23	01		UPC1:TxData[1] UTOPIA 8[1] UPC1:TxData[9] UTOPIA 16[1] UPC1:TDAT[9] POS	01		TDMd:CLKO	01				
PB24	IN	GPI_PB24	10		UCC4:COL Enet	10	GND	CE:EXT_REQ1	10	GND			
	OUT	GPO_PB24	01		UPC1:TxData[0] UTOPIA 8[1] UPC1:TxData[8] UTOPIA 16[1] UPC1:TDAT[8] POS	01		SI:STRB4	01				
PB25	IN	GPI_PB25	10		UCC4:CRS Enet	10	GND				TDMa:RxD[0]/ TDMa:RxD (Serial) (Primary Option)	11	PA8
	OUT	GPO_PB25	01		UPC1:TxData[7] UTOPIA 16[1] UPC1:TDAT[7] POS	01		TDMd:$\overline{RQ}$	01				

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

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

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

Table 3-14. Port C Dedicated Pin Assignment

Pin	Direction	Pin Functions											
		CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC0	IN	GPI_PC0	10		CLK1	10	GND	Timer1:TIN1 Timer2:TIN2 (Secondary Option)	10	GND			
	OUT	GPO_PC0	01		$\overline{\text{DMA\_DACK3}}$ (Primary Option)	01		PTP_ALARM1 (Primary Option)	01		BRGO1	01	
PC1	IN	GPI_PC1	10		CLK2	10	GND	Timer1:$\overline{\text{TGATE1}}$ Timer2:$\overline{\text{TGATE2}}$ (Secondary Option)	10	VCC			
	OUT	GPO_PC1	01					$\overline{\text{DMA\_DACK0}}$ (Secondary Option)	01		BRGO2	01	
PC2	IN	GPI_PC2	10		CLK3	10	GND						
	OUT	GPO_PC2	01					UCC2:GTx Clock Enet	01		UCC2:CLKO	01	
PC3	IN	GPI_PC3	10		CLK4	10	GND	Timer1:$\overline{\text{TGATE4}}$ Timer2:$\overline{\text{TGATE3}}$ (Third Option)	10	VCC			
	OUT	GPO_PC3	01		SI:STRB1	01		UCC4:CLKO	01		BRGO3	01	

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

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC4	IN	GPI_PC4	10		CLK5	10	GND						
	OUT	GPO_PC4	01		SI:STRB8	01		UCC8:CLKO	01		BRGO4	01	
PC5	IN	GPI_PC5	10		CLK6	10	GND						
	OUT	GPO_PC5	01					BRGO5	01		PTP_PPS3 (Primary Option)	01	
PC6	IN	GPI_PC6	10		CLK7	10	GND						
	OUT	GPO_PC6	01								UPC2:CLKO	01	
PC7	IN	GPI_PC7	10		CLK8	10	GND						
	OUT	GPO_PC7	01					UCC6:CLKO	01		BRGO6	01	
PC8	IN	GPI_PC8	10		CLK9	10	GND						
	OUT	GPO_PC8	01								BRGO7	01	
PC9	IN	GPI_PC9	10		CLK10	10	GND						
	OUT	GPO_PC9	01					UCC1:CLKO	01		UCC1:GTx Clock Enet	01	
PC10	IN	GPI_PC10	10		CLK11	10	GND	UPC2:RMOD POS master UPC2:TMOD POS slave (Secondary Option)	10	GND	UCC1:COL Enet (Secondary Option)	10	PA13
	OUT	GPO_PC10	01		Timer1:$\overline{\text{TOUT4}}$	01		SI:STRB2	01		UCC3:CLKO	01	
PC11	IN	GPI_PC11	10		CLK12	10	GND	Timer1:TIN4 Timer2:TIN3 1588_EXT_TRIG2 (Secondary Option)	10	GND	UCC2:COL Enet (Secondary Option)	10	PA27
	OUT	GPO_PC11	01		$\overline{\text{DMA\_DDONE2}}$ (Primary Option)	01		UPC2:TERR POS master UPC2:RERR POS slave	01		BRGO8	01	
PC12	IN	GPI_PC12	10		CLK13	10	GND	CE:EXT_REQ2	10	GND			
	OUT	GPO_PC12	01					PTP_REF_CLK (Primary Option)	01		UPC1:CLKO	01	
PC13	IN	GPI_PC13	10		CLK14	10	GND	$\overline{\text{DMA\_DREQ0}}$ (Secondary Option)	10	VCC			
	OUT	GPO_PC13	01		UPC1:$\overline{\text{TxEnb}}$[3] UTOPIA UPC1:TENB[3] POS	01		UCC5:CLKO	01				

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

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC14	IN	GPI_PC14	10		CLK15	10	GND	Timer1:TIN3 Timer2:TIN4 1588_EXT_TRIG1 (Primary Option)	10	PB4	UPC2:REOP POS master UPC2:TEOP POS slave (Secondary Option)	10	GND
	OUT	GPO_PC14	01					UCC7:CLKO	01		BRGO9	01	
PC15	IN	GPI_PC15	10		CLK16	10	GND	Timer1:$\overline{\text{TGATE4}}$ Timer2:$\overline{\text{TGATE3}}$ (Secondary Option)	10	PC3	Timer1:$\overline{\text{TGATE2}}$ Timer2:$\overline{\text{TGATE1}}$ (Primary Option)	10	PA31
	OUT	GPO_PC15	01								BRGO10	01	
PC16	IN	GPI_PC16	10		CLK17	10	GND	UPC1:TxClav[3] UTOPIA UPC1:PTPA[3]/DTPA[3] POS (Secondary Option)	10	GND			
	OUT	GPO_PC16	01		$\overline{\text{DMA\_DACK2}}$ (Primary Option)	01					BRGO11	01	
PC17	IN	GPI_PC17	10		CLK18	10	GND	$\overline{\text{DMA\_DREQ3}}$ (Secondary Option)	10	VCC	UPC1:RxClav[3] UTOPIA UPC1:PRPA[3]/ DRPA[3] POS (Secondary Option)	10	GND
	OUT	GPO_PC17	01					BRGO12	01		PTP_PPS2 (Primary Option)	01	
PC18	IN	GPI_PC18	10		CLK19	10	GND						
	OUT	GPO_PC18	01		$\overline{\text{DMA\_DDONE3}}$ (Primary Option)	01		BRGO13	01		Timer2:$\overline{\text{TOUT3}}$	01	
PC19	IN	GPI_PC19	10		CLK20	10	GND						
	OUT	GPO_PC19	01					UPC1:$\overline{\text{RxEnb}}$[3] UTOPIA UPC1:RENB[3] POS	01		BRGO14	01	
PC20	IN	GPI_PC20	10		CLK21	10	GND						
	OUT	GPO_PC20	01		UPC2:TEOP POS master UPC2:REOP POS slave	01		UPC1:$\overline{\text{TxEnb}}$[2] UTOPIA UPC1:TENB[2] POS	01		BRGO15	01	
PC21	IN	GPI_PC21	10		CLK22	10	GND						
	OUT	GPO_PC21	01					PTP_PPS1 (Primary Option)	01		UPC1:$\overline{\text{RxEnb}}$[2] UTOPIA UPC1:RENB[2] POS	01	

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

Pin	Pin Functions												
	Direction	CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC22	IN	GPI_PC22	10		CLK23	10	GND				$\overline{\text{DMA\_DREQ3}}$ (Primary Option)	10	PC17
	OUT	GPO_PC22	01		UPC1:TMOD POS master UPC1:RMOD POS slave	01		UCC1:TxD[7]/TCG[7] Enet UCC1:TxD[7] SER	01		UPC2:TxAddr[5] UTOPIA UPC2:TADR[5] POS	01	
PC23	IN	GPI_PC23	10		CLK24	10	GND	UPC1:RMOD POS master UPC1:TMOD POS slave (Secondary Option)	10	GND	$\overline{\text{DMA\_DREQ0}}$ (Primary Option)	10	PC13
	OUT	GPO_PC23	01		PTP_ALARM2 (Primary Option)	01		UCC1:TxD[6]/TCG[6] Enet UCC1:TxD[6] SER	01				
PC24	IN	GPI_PC24	10		UPC1:REOP POS master UPC1:TEOP POS slave	10	GND	UPC2:TxAddr[4] UTOPIA slave UPC2:TADR[4] POS slave (Secondary Option)	10	GND			
	OUT	GPO_PC24	01		UCC1:TxD[5]/TCG[5] Enet UCC1:TxD[5] SER	01		UPC2:RxAddr[4] UTOPIA master UPC2:RADR[4] POS master	01		Timer1:$\overline{\text{TOUT3}}$	01	
PC25	IN	GPI_PC25	10		Timer1:$\overline{\text{TGATE3}}$ Timer2:$\overline{\text{TGATE4}}$ (Secondary Option)	10	VCC	UPC2:RxData[5] UTOPIA 8[1] UPC2:RxData[13] UTOPIA 16[1] UPC2:RDAT[13][1] POS (Secondary Option)	10	GND			
	OUT	GPO_PC25	01		BRGO16	01		UCC1:TxD[4]/TCG[4] Enet UCC1:TxD[4] SER	01		UPC1:TEOP POS master UPC1:REOP POS slave	01	
PC26	IN	GPI_PC26	10		RTC_CLK/PIT_CLK	10	GND						
	OUT	GPO_PC26	01					SI:STRB3	01				
PC27	IN	GPI_PC27	10		Timer1:TIN4 Timer2:TIN3 1588_EXT_TRIG2 (Primary Option)	10	PC11	TDMb:RxD[0]/ TDMb:RxD (Serial) (Primary Option)	11	PA27	TDMf:RSYNC (Secondary Option)	10	PA23
	OUT	GPO_PC27	01		SI:STRB4	01							
PC28	IN	GPI_PC28	10		UCC2:TBI RX_CLK1 Enet (Secondary Option)	10	GND	TDMb:TxD[0]/ TDMb:TxD (Serial) (Primary Option)	11	PA22	TDMf:TSYNC/GRANT (Secondary Option)	10	PA21
	OUT	GPO_PC28	01		Timer1:$\overline{\text{TOUT4}}$	01					SI:STRB5	01	

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

Pin	Direction	Pin Functions											
		CPPARCx[SELn]=00			CPPARCx[SELn]=01			CPPARCx[SELn]=10			CPPARCx[SELn]=11		
		Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input	Function	CPDIRxC[DIRn]	Default Input
PC29	IN	GPI_PC29	10		UCC1:TBI RX_CLK1 Enet	10	GND	TDMb:RSYNC (Primary Option)	10	PA28	Timer1:$\overline{\text{TGATE4}}$ Timer2:$\overline{\text{TGATE3}}$ (Primary Option)	10	PC15
	OUT	GPO_PC29	01		Timer2:$\overline{\text{TOUT3}}$	01		SI:STRB6	01		$\overline{\text{DMA\_DACK0}}$ (Primary Option)	01	
PC30	IN	GPI_PC30	10		PTP_REF_CLK	10	GND						
	OUT	GPO_PC30	01										

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

Table 3-15. Port D Dedicated Pin Assignment

Pin	Direction	Pin Functions											
		CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD0	IN	GPI_PD0	10										
	OUT	GPO_PD0	01		UCC5:TxD[0] Enet UCC5:TxD[0]/ UCC5:TxD (Serial) SER	01		TDMe:TxD[1]	01		UPC1:TxData[4] UTOPIA 16[1] UPC1:TDAT[4] POS	01	
PD1	IN	GPI_PD1	10										
	OUT	GPO_PD1	01		UCC5:TxD[1] Enet UCC5:TxD[1] SER	01		TDMe:TxD[2]	01		UPC1:TxData[3] UTOPIA 16[1] UPC1:TDAT[3] POS	01	
PD2	IN	GPI_PD2	10					TDMe:RSYNC (Secondary Option)	10	PA26			
	OUT	GPO_PD2	01		UCC5:TxD[2] Enet UCC5:TxD[2] SER	01		UPC1:TxData[2] UTOPIA 16[1] UPC1:TDAT[2] POS	01				
PD3	IN	GPI_PD3	10					TDMe:TSYNC/GRANT (Secondary Option)	10	PA17	TDMg:TxD[0]/ TDMg:TxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_PD3	01		UCC5:TxD[3] Enet UCC5:TxD[3] SER	01		UPC1:TxData[1] UTOPIA 16 UPC1:TDAT[1] POS	01				

Table 3-15. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD4	IN	GPI_PD4	10										
	OUT	GPO_PD4	01		UCC5:TX_EN Enet UCC5:$\overline{\text{RTS}}$ SER	01					UPC1:TxData[0] UTOPIA 16[1] UPC1:TDAT[0] POS	01	
PD5	IN	GPI_PD5	10		UPC1:RxSOC UTOPIA master UPC1:TxSOC UTOPIA slave UPC1:RSOP POS master UPC1:TSOP POS slave	10	GND	TDMe:TxD[0]/ TDMe:TxD (Serial) (Secondary Option)	11	PA30			
	OUT	GPO_PD5	01		UCC5:TX_ER Enet	01							
PD6	IN	GPI_PD6	10		UCC5:RxD[0] Enet UCC5:RxD[0]/ UCC5:RxD (Serial) SER (Secondary Option)	10	GND	TDMe:RxD[1]	10	GND	UPC1:$\overline{\text{TxEnb}}$ UTOPIA slave UPC1:TENB POS slave	10	VCC
	OUT	GPO_PD6	01								UPC1:$\overline{\text{RxEnb}}$[0] UTOPIA master UPC1:RENB[0] POS master	01	
PD7	IN	GPI_PD7	10		UCC5:RxD[1] Enet UCC5:RxD[1] SER (Secondary Option)	10	GND	TDMe:RxD[2]	10	GND	UPC1:RxClav[0] UTOPIA master UPC1:PRPA[0]/ DRPA[0] POS master	10	GND
	OUT	GPO_PD7	01								UPC1:TxClav UTOPIA slave UPC1:PTPA/DTPA POS slave	01	
PD8	IN	GPI_PD8	10		UCC5:RxD[2] Enet UCC5:RxD[2] SER	10	GND	TDMe:RxD[0]/ TDMe:RxD (Serial) (Secondary Option)	11	PA29	UPC1:RxData[7] UTOPIA 8[1] UPC1:RxData[15] UTOPIA 16[1] UPC1:RDAT[15] POS	10	GND
	OUT	GPO_PD8	01										
PD9	IN	GPI_PD9	10		UCC5:RxD[3] Enet UCC5:RxD[3] SER	10	GND	UPC1:RxData[6] UTOPIA 8[1] UPC1:RxData[14] UTOPIA 16[1] UPC1:RDAT[14] POS	10	GND			
	OUT	GPO_PD9	01					TDMe:CLKO	01				

Table 3-15. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD10	IN	GPI_PD10	10		UCC5:COL Enet	10	GND				UPC1:RxData[5] UTOPIA 8[1] UPC1:RxData[13] UTOPIA 16[1] UPC1:RDAT[13] POS	10	GND
	OUT	GPO_PD10	01					SI:STRB5	01				
PD11	IN	GPI_PD11	10		UCC5:CRS Enet	10	GND				UPC1:RxData[4] UTOPIA 8[1] UPC1:RxData[12] UTOPIA 16[1] UPC1:RDAT[12] POS	10	GND
	OUT	GPO_PD11	01					TDMe:$\overline{RQ}$	01				
PD12	IN	GPI_PD12	10		UCC5:RX_DV Enet UCC5:$\overline{CTS}$ SER (Secondary Option)	10	GND	UPC1:RxData[3] UTOPIA 8[1] UPC1:RxData[11] UTOPIA 16[1] UPC1:RDAT[11] POS	10	GND			
	OUT	GPO_PD12	01					TDMe:TxD[3]	01				
PD13	IN	GPI_PD13	10		UCC5:RX_ER Enet UCC5:$\overline{CD}$ SER (Secondary Option)	10	GND	TDMe:RxD[3]	10	GND	UPC1:RxData[2] UTOPIA 8[1] UPC1:RxData[10] UTOPIA 16[1] UPC1:RDAT[10][1] POS	10	GND
	OUT	GPO_PD13	01										
PD14	IN	GPI_PD14	10					CE:EXT_REQ3	10	GND	UPC1:RxData[1] UTOPIA 8[1] UPC1:RxData[9] UTOPIA 16[1] UPC1:RDAT[9] POS	10	GND
	OUT	GPO_PD14	01		UCC6:TxD[0] Enet UCC6:TxD[0]/ UCC6:TxD (Serial) SER	01		TDMf:TxD[1]	01				
PD15	IN	GPI_PD15	10					CE:EXT_REQ4	10	GND	UPC1:RxData[0] UTOPIA 8[1] UPC1:RxData[8] UTOPIA 16[1] UPC1:RDAT[8] POS	10	GND
	OUT	GPO_PD15	01		UCC6:TxD[1] Enet UCC6:TxD[1] SER	01		TDMf:TxD[2]	01				

Table 3-15. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD16	IN	GPI_PD16	10		TDMf:RSYNC (Primary Option)	10	PC27	Timer1:TIN2 Timer2:TIN1 (Secondary Option)	10	GND	UPC2:RxAddr[2] UTOPIA slave UPC2:RADR[2] POS slave (Secondary Option)	10	GND
	OUT	GPO_PD16	01		UCC6:TxD[2] Enet UCC6:TxD[2] SER	01		UPC1:TERR POS master UPC1:RERR POS slave	01		UPC2:TxAddr[2] UTOPIA master UPC2:TADR[2] POS master	01	
PD17	IN	GPI_PD17	10		UPC1:RERR POS master UPC1:TERR POS slave	10	GND	TDMf:TSYNC/GRANT (Primary Option)	10	PC28	UPC2:RxAddr[1] UTOPIA slave UPC2:RADR[1] POS slave (Secondary Option)	10	GND
	OUT	Timer2:$\overline{\text{TOUT1}}$	01		UCC6:TxD[3] Enet UCC6:TxD[3] SER	01		Timer1:$\overline{\text{TOUT2}}$	01		UPC2:TxAddr[1] UTOPIA master UPC2:TADR[1] POS master	01	
PD18	IN	GPI_PD18	10					UPC1:RxData[7] UTOPIA 16[1] UPC1:RDAT[7] POS	10	GND			
	OUT	GPO_PD18	01		UCC6:TX_EN Enet UCC6:$\overline{\text{RTS}}$ SER	01							
PD19	IN	GPI_PD19	10		$\overline{\text{DMA\_DREQ2}}$ (Primary Option)	10	PB5	TDMf:RxD[0]/ TDMf:RxD (Serial) (Secondary Option)	11	GND	UPC1:RxPrty UTOPIA master UPC1:TxPrty UTOPIA slave UPC1:RPRTY POS master UPC1:TPRTY POS slave	10	GND
	OUT	GPO_PD19	01		UCC6:TX_ER Enet	01					PTP_PPS2 (Secondary Option)	01	
PD20	IN	GPI_PD20	10		UCC6:RxD[0] Enet UCC6:RxD[0]/ UCC6:RxD (Serial) SER	10	GND	TDMf:RxD[1]	10	GND	UPC1:RxData[6] UTOPIA 16[1] UPC1:RDAT[6] POS	10	GND
	OUT	GPO_PD20	01										
PD21	IN	GPI_PD21	10		UCC6:RxD[1] Enet UCC6:RxD[1] SER	10	GND	TDMf:RxD[2]	10	GND	UPC1:RxData[5] UTOPIA 16[1] UPC1:RDAT[5] POS	10	GND
	OUT	GPO_PD21	01										

Table 3-15. Port D Dedicated Pin Assignment (continued)

Pin	Pin Functions												
	Direction	CPPARDx[SELn]=00			CPPARDx[SELn]=01			CPPARDx[SELn]=10			CPPARDx[SELn]=11		
		Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input	Function	CPDIRxD[DIRn]	Default Input
PD22	IN	GPI_PD22	10		UCC6:RxD[2] Enet UCC6:RxD[2] SER	10	GND	TDMf:TxD[0]/ TDMf:TxD (Serial) (Secondary Option)	11	GND			
	OUT	GPO_PD22	01		UPC1:TxPrty UTOPIA master UPC1:RxPrty UTOPIA slave UPC1:TPRTY POS master UPC1:RPRTY POS slave	01							
PD23	IN	GPI_PD23	10		UCC6:RxD[3] Enet UCC6:RxD[3] SER	10	GND	TDMg:TSYNC/GRANT (Secondary Option)	10	GND	UPC1:RxData[4] UTOPIA 16[1] UPC1:RDAT[4] POS	10	GND
	OUT	GPO_PD23	01					TDMf:CLKO	01				
PD24	IN	GPI_PD24	10		UCC6:COL Enet	10	GND	UPC1:RxData[3] UTOPIA 16[1] UPC1:RDAT[3] POS	10	GND	TDMg:RxD[0]/ TDMg:RxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_PD24	01					SI:STRB6	01				
PD25	IN	GPI_PD25	10		UCC6:CRS Enet	10	GND	UPC1:RxData[2] UTOPIA 16[1] UPC1:RDAT[2] POS	10	GND	TDMg:RSYNC (Secondary Option)	10	GND
	OUT	GPO_PD25	01		TDMf:$\overline{RQ}$	01							
PD26	IN	GPI_PD26	10		UCC6:RX_DV Enet UCC6:$\overline{CTS}$ SER	10	GND	UPC1:RxData[1] UTOPIA 16[1] UPC1:RDAT[1] POS	10	GND			
	OUT	GPO_PD26	01					TDMf:TxD[3]	01				
PD27	IN	GPI_PD27	10		UCC6:RX_ER Enet UCC6:$\overline{CD}$ SER	10	GND	TDMf:RxD[3]	10	GND	UPC1:RxData[0] UTOPIA 16[1] UPC1:RDAT[0] POS	10	GND
	OUT	GPO_PD27	01										

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

Table 3-16. Port E Dedicated Pin Assignment

Pin	Pin Functions											
	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE0	IN	GPI_PE0	10					TDMh:TxD[0]/ TDMh:TxD (Serial) (Secondary Option)	11	GND			
	OUT	GPO_PE0	01		UCC7:TxD[0] Enet UCC7:TxD[0]/ UCC7:TxD (Serial) SER	01					UPC1:RxAddr[5] UTOPIA UPC1:RADR[5] POS	01	
PE1	IN	GPI_PE1	10					TDMc:TSYNC/GRANT (Primary Option)	10	PB3	UPC1:RxAddr[3] UTOPIA slave UPC1:RADR[3] POS slave	10	GND
	OUT	GPO_PE1	01		UCC7:TxD[1] Enet UCC7:TxD[1] SER	01		UPC1:TxAddr[3] UTOPIA master UPC1:TADR[3] POS master	01				
PE2	IN	GPI_PE2	10		TDMg:RSYNC (Primary Option)	10	PD25	UPC1:TxAddr[0] UTOPIA slave UPC1:TADR[0] POS slave	10	GND			
	OUT	GPO_PE2	01		UCC7:TxD[2] Enet UCC7:TxD[2] SER	01		UPC1:RxAddr[0] UTOPIA master UPC1:RADR[0] POS master	01				
PE3	IN	GPI_PE3	10		TDMg:TSYNC/GRANT (Primary Option)	10	PD23	UPC1:RxAddr[0] UTOPIA slave UPC1:RADR[0] POS slave	10	GND			
	OUT	GPO_PE3	01		UCC7:TxD[3] Enet UCC7:TxD[3] SER	01		UPC1:TxAddr[0] UTOPIA master UPC1:TADR[0] POS master	01				
PE4	IN	GPI_PE4	10					UPC1:TxAddr[2] UTOPIA slave UPC1:TADR[2] POS slave	10	GND	TDMh:TSYNC/GRANT (Secondary Option)	10	GND
	OUT	GPO_PE4	01		UCC7:TX_EN Enet UCC7:$\overline{\text{RTS}}$ SER	01		UPC1:RxAddr[2] UTOPIA master UPC1:RADR[2] POS master	01				
PE5	IN	GPI_PE5	10					TDMg:TxD[0]/ TDMg:TxD (Serial) (Primary Option)	11	PD3	UPC1:TxAddr[1] UTOPIA slave UPC1:TADR[1] POS slave	10	GND
	OUT	GPO_PE5	01		UCC7:TX_ER Enet	01					UPC1:RxAddr[1] UTOPIA master UPC1:RADR[1] POS master	01	

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

Pin	Direction	Pin Functions											
		CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE6	IN	GPI_PE6	10		UCC7:RxD[0] Enet UCC7:RxD[0]/ UCC7:RxD (Serial) SER	10	GND	TDMh:RSYNC (Secondary Option)	10	GND			
	OUT	GPO_PE6	01		PTP_ALARM2 (Secondary Option)	01					UPC1:TxAddr[5] UTOPIA UPC1:TADR[5] POS	01	
PE7	IN	GPI_PE7	10		UCC7:RxD[1] Enet UCC7:RxD[1] SER	10	GND	UPC1:TxAddr[4] UTOPIA slave UPC1:TADR[4] POS slave	10	GND	TDMc:TxD[0]/ TDMc:TxD (Serial) (Primary Option)	11	PB8
	OUT	GPO_PE7	01					UPC1:RxAddr[4] UTOPIA master UPC1:RADR[4] POS master	01				
PE8	IN	GPI_PE8	10		UCC7:RxD[2] Enet UCC7:RxD[2] SER	10	GND	TDMg:RxD[0]/ TDMg:RxD (Serial) (Primary Option)	11	PD24	UPC1:RxAddr[1] UTOPIA slave UPC1:RADR[1] POS slave	10	GND
	OUT	GPO_PE8	01								UPC1:TxAddr[1] UTOPIA master UPC1:TADR[1] POS master	01	
PE9	IN	GPI_PE9	10		UCC7:RxD[3] Enet UCC7:RxD[3] SER	10	GND	TDMf:RxD[0]/ TDMf:RxD (Serial) (Primary Option)	11	PD19	UPC1:STPA POS master	10	GND
	OUT	GPO_PE9	01		TDMg:CLKO	01					UPC1:STPA POS slave	01	
PE10	IN	GPI_PE10	10		UCC7:COL Enet	10	GND	TDMf:TxD[0]/ TDMf:TxD (Serial) (Primary Option)	11	PD22	UPC1:RVAL POS master (Secondary Option)	10	GND
	OUT	GPO_PE10	01		SI:STRB7	01					UPC1:RVAL POS slave	01	
PE11	IN	GPI_PE11	10		UCC7:CRS Enet	10	GND	UPC1:TxAddr[3] UTOPIA slave UPC1:TADR[3] POS slave	10	GND	TDMc:RxD[0]/ TDMc:RxD (Serial) (Primary Option)	11	PB5
	OUT	GPO_PE11	01		UPC1:RxAddr[3] UTOPIA master UPC1:RADR[3] POS master	01		TDMg:$\overline{RQ}$	01				

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

Pin	Pin Functions											
	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE12	IN	GPI_PE12	10		UCC7:RX_DV Enet UCC7:$\overline{\text{CTS}}$ SER	10	GND	UPC1:RxAddr[2] UTOPIA slave UPC1:RADR[2] POS slave	10	GND	TDMh:RxD[0]/ TDMh:RxD (Serial) (Secondary Option)	11	GND
	OUT	GPO_PE12	01					UPC1:TxAddr[2] UTOPIA master UPC1:TADR[2] POS master	01				
PE13	IN	GPI_PE13	10		UCC7:RX_ER Enet UCC7:$\overline{\text{CD}}$ SER	10	GND	UPC1:RxAddr[4] UTOPIA slave UPC1:RADR[4] POS slave	10	GND	TDMc:RSYNC (Primary Option)	10	PB2
	OUT	GPO_PE13	01								UPC1:TxAddr[4] UTOPIA master UPC1:TADR[4] POS master	01	
PE14	IN	GPI_PE14	10					TDMe:RSYNC (Primary Option)	10	PD2	UPC2:RxData[0] UTOPIA 8[1] UPC2:RxData[8] UTOPIA 16[1] UPC2:RDAT[8] POS (Secondary Option)	10	GND
	OUT	GPO_PE14	01		PTP_ALARM1 (Secondary Option)	01		UCC8:TxD[0] Enet UCC8:TxD[0]/ UCC8:TxD (Serial) SER	01		TDMh:TxD[1]	01	
PE15	IN	GPI_PE15	10		TDMe:TSYNC/GRANT (Primary Option)	10	PD3	TDMh:RxD[2]	10	GND	UPC2:$\overline{\text{RxEnb}}$ UTOPIA slave UPC2:RENB POS slave (Secondary Option)	10	VCC
	OUT	GPO_PE15	01		UCC8:TxD[1] Enet UCC8:TxD[1] SER	01		UPC1:$\overline{\text{TxEnb}}$[2] UTOPIA UPC1:TENB[2] POS	01		UPC2:$\overline{\text{TxEnb}}$[0] UTOPIA master UPC2:TENB[0] POS master	01	
PE16	IN	GPI_PE16	10		UPC1:RxClav[1] UTOPIA UPC1:PRPA[1]/ DRPA[1] POS (Primary Option)	10	PA8	TDMh:RSYNC (Primary Option)	10	PE6	UPC2:TxClav[0] UTOPIA master UPC2:PTPA[0]/DTPA[0] POS master (Secondary Option)	10	GND
	OUT	GPO_PE16	01		UCC8:TxD[2] Enet UCC8:TxD[2] SER	01		UCC5:TxD[0] Enet UCC5:TxD[0]/ UCC5:TxD (Serial) SER	01		UPC2:RxClav UTOPIA slave UPC2:PRPA/DRPA POS slave	01	

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

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE17	IN	GPI_PE17	10		UPC1:TxClav[1] UTOPIA UPC1:PTPA[1]/DTPA[1] POS (Primary Option)	10	PA13	TDMh:TSYNC/GRANT (Primary Option)	10	PE4	UART2:SIN (Secondary Option)	10	VCC
	OUT	GPO_PE17	01		UCC8:TxD[3] Enet UCC8:TxD[3] SER	01		UPC2:TxData[7] UTOPIA 8[1] UPC2:TxData[15] UTOPIA 16[1] UPC2:TDAT[15] POS	01		UCC5:TxD[1] Enet UCC5:TxD[1] SER	01	
PE18	IN	GPI_PE18	10								TDMe:TxD[0]/ TDMe:TxD (Serial) (Primary Option)	11	PD5
	OUT	GPO_PE18	01		UCC8:TX_EN Enet UCC8:$\overline{\text{RTS}}$ SER	01		UPC1:$\overline{\text{RxEnb}}$[3] UTOPIA UPC1:RENB[3] POS	01				
PE19	IN	GPI_PE19	10		UPC2:RxData[2] UTOPIA 8[1] UPC2:RxData[10] UTOPIA 16[1] UPC2:RDAT[10][1] POS (Secondary Option)	10	GND	TDMh:RxD[0]/ TDMh:RxD (Serial) (Primary Option)	11	PE12	UCC5:RxD[1] Enet UCC5:RxD[1] SER (Primary Option)	10	PD7
	OUT	GPO_PE19	01		UCC8:TX_ER Enet	01					UPC1:$\overline{\text{RxEnb}}$[1] UTOPIA UPC1:RENB[1] POS	01	
PE20	IN	GPI_PE20	10		UCC8:RxD[0] Enet UCC8:RxD[0]/ UCC8:RxD (Serial) SER	10	GND	TDMh:RxD[1]	10	GND	UPC2:RxAddr[4] UTOPIA slave UPC2:RADR[4] POS slave (Secondary Option)	10	GND
	OUT	GPO_PE20	01					PTP_PPS1 (Secondary Option)	01		UPC2:TxAddr[4] UTOPIA master UPC2:TADR[4] POS master	01	
PE21	IN	GPI_PE21	10		UCC8:RxD[1] Enet UCC8:RxD[1] SER	10	GND	UPC1:RxClav[3] UTOPIA UPC1:PRPA[3]/ DRPA[3] POS (Primary Option)	10	PC17	TDMe:RxD[0]/ TDMe:RxD (Serial) (Primary Option)	11	PD8
	OUT	GPO_PE21	01		TDMh:TxD[2]	01		UPC2:TxData[3] UTOPIA 8[1] UPC2:TxData[11] UTOPIA 16[1] UPC2:TDAT[11] POS	01				

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

Pin	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE22	IN	GPI_PE22	10		UCC8:RxD[2] Enet UCC8:RxD[2] SER	10	GND	UCC5:RxD[0] Enet UCC5:RxD[0]/ UCC5:RxD (Serial) SER (Primary Option)	10	PD6	TDMh:TxD[0]/ TDMh:TxD (Serial) (Primary Option)	11	PE0
	OUT	GPO_PE22	01		UPC1:$\overline{\text{TxEnb}}$[1] UTOPIA UPC1:TENB[1] POS	01		UPC2:TxData[5] UTOPIA 8[1] UPC2:TxData[13] UTOPIA 16[1] UPC2:TDAT[13] POS	01				
PE23	IN	GPI_PE23	10		UCC8:RxD[3] Enet UCC8:RxD[3] SER	10	GND	UPC1:RxClav[2] UTOPIA UPC1:PRPA[2]/ DRPA[2] POS (Primary Option)	10	PA22	UART2:$\overline{\text{CTS}}$ (Secondary Option)	10	GND
	OUT	GPO_PE23	01		TDMh:CLKO	01		UPC2:TxData[1] UTOPIA 8[1] UPC2:TxData[9] UTOPIA 16[1] UPC2:TDAT[9] POS	01		UCC5:TX_EN Enet UCC5:$\overline{\text{RTS}}$ SER	01	
PE24	IN	GPI_PE24	10		UCC8:COL Enet	10	GND	UPC1:TxClav[2] UTOPIA UPC1:PTPA[2]/DTPA[2] POS (Primary Option)	10	PA27	UCC5:RX_DV Enet UCC5:$\overline{\text{CTS}}$ SER (Primary Option)	10	PD12
	OUT	GPO_PE24	01		SI:STRB8	01		UART2:SOUT	01		UPC2:TxData[0] UTOPIA 8[1] UPC2:TxData[8] UTOPIA 16[1] UPC2:TDAT[8] POS	01	
PE25	IN	GPI_PE25	10		UCC8:CRS Enet	10	GND	UPC2:RxSOC UTOPIA master UPC2:TxSOC UTOPIA slave UPC2:RSOP POS master UPC2:TSOP POS slave (Secondary Option)	10	GND	UCC5:RX_ER Enet UCC5:$\overline{\text{CD}}$ SER (Primary Option)	10	PD13
	OUT	GPO_PE25	01		UPC1:$\overline{\text{RxEnb}}$[2] UTOPIA UPC1:RENB[2] POS	01		TDMh:$\overline{\text{RQ}}$	01		UART2:$\overline{\text{RTS}}$	01	

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

Pin	Pin Functions												
	Direction	CPPAREx[SELn]=00			CPPAREx[SELn]=01			CPPAREx[SELn]=10			CPPAREx[SELn]=11		
		Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input	Function	CPDIRxEDIRn]	Default Input	Function	CPDIRxE[DIRn]	Default Input
PE26	IN	GPI_PE26	10		UCC8:RX_DV Enet UCC8:$\overline{\text{CTS}}$ SER	10	GND	TDMh:RxD[3]	10	GND	UPC2:$\overline{\text{TxEnb}}$ UTOPIA slave UPC2:TENB POS slave (Secondary Option)	10	VCC
	OUT	GPO_PE26	01		PTP_REF_CLK (Secondary Option)	01		UPC1:$\overline{\text{TxEnb}}$[3] UTOPIA UPC1:TENB[3] POS	01		UPC2:$\overline{\text{RxEnb}}$[0] UTOPIA master UPC2:RENB[0] POS master	01	
PE27	IN	GPI_PE27	10		UCC8:RX_ER Enet UCC8:$\overline{\text{CD}}$ SER	10	GND	UPC1:TxClav[3] UTOPIA UPC1:PTPA[3]/DTPA[3] POS (Primary Option)	10	PC16	UPC2:RxClav[0] UTOPIA master UPC2:PRPA[0]/ DRPA[0] POS master (Secondary Option)	10	GND
	OUT	GPO_PE27	01		DMA_DDONE1 (Secondary Option)	01		TDMh:TxD[3]	01		UPC2:TxClav UTOPIA slave UPC2:PTPA/ DTPA POS slave	01	
PE28	IN	GPI_PE28	10								SPI:SPIMOSI	11	VCC
	OUT	GPO_PE28	01										
PE29	IN	GPI_PE29	10								SPI:SPIMISO	11	SPI-MOSI
	OUT	GPO_PE29	01										
PE30	IN	GPI_PE30	10								SPI:SPICLK	11	GND
	OUT	GPO_PE30	01										
PE31	IN	GPI_PE31	10		Timer1: TIN2 Timer2: TIN1 (Primary Option)	10	PD16				SPI: $\overline{\text{SPISEL}}$	10	VCC
	OUT	GPO_PE31	01		SI: STRB7	01		Timer1: $\overline{\text{TOUT3}}$	01				

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

Table 3-17. Port F Dedicated Pin Assignment

Pin	Pin Functions												
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF0	IN	GPI_PF0	10		Timer1:$\overline{\text{TGATE1}}$ Timer2:$\overline{\text{TGATE2}}$ (Primary Option)	10	PC1	UPC2:RxData[5] UTOPIA 16[1] UPC2:RDAT[5] POS	10	GND	UPC1:RMOD POS master UPC1:TMOD POS slave (Primary Option)	10	PC23
	OUT	GPO_PF0	01					UART2:SOUT	01		DMA_DACK1 (Secondary Option)	01	
PF1	IN	GPI_PF1	10		UPC2:RxData[1] UTOPIA 16[1] UPC2:RDAT[1] POS	10	GND	Timer1:TIN1 Timer2:TIN2 (Primary Option)	10	PC0	UART2:$\overline{\text{CTS}}$ (Primary Option)	10	PE23
	OUT	GPO_PF1	01					UPC1:TMOD POS master UPC1:RMOD POS slave	01		DMA_DDONE1 (Primary Option)	01	
PF2	IN	GPI_PF2	10					UPC2:RxData[0] UTOPIA 16[1] UPC2:RDAT[0] POS	10	GND	UPC1:RVAL POS master (Primary Option)	10	PE10
	OUT	GPO_PF2	01		UART2:$\overline{\text{RTS}}$	01		UPC1:RVAL POS slave	01		Timer1:$\overline{\text{TOUT1}}$	01	
PF3	IN	GPI_PF3	10		DMA_DREQ1 (Secondary Option)	10	VCC	UART2:SIN (Primary Option)	10	PE17	UPC2:RxData[1] UTOPIA 8[1] UPC2:RxData[9] UTOPIA 16[1] UPC2:RDAT[9] POS (Primary Option)	10	PB1
	OUT	GPO_PF3	01		Timer1:$\overline{\text{TOUT2}}$	01					Timer2:$\overline{\text{TOUT1}}$	01	
PF4	IN	GPI_PF4	10					UPC2:RxData[6] UTOPIA 16[1] UPC2:RDAT[6] POS	10	GND	M66EN	10	VCC
	OUT	GPO_PF4	01										
PF5	IN	GPI_PF5	10		UPC2:TxAddr[0] UTOPIA slave UPC2:TADR[0] POS slave (Primary Option)	10	PB0						
	OUT	GPO_PF5	01		UPC2:RxAddr[0] UTOPIA master UPC2:RADR[0] POS master	01		UCC2:TxD[0]/TCG[0] Enet UCC2:TxD[0]/ UCC2:TxD (Serial) SER	01		$\overline{\text{PCI\_INTA/IRQ\_OUT}}$	01	

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

Pin	Pin Functions											
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF6	IN	GPI_PF6	10		UPC2:TxAddr[1] UTOPIA slave UPC2:TADR[1] POS slave (Primary Option)	10	PA8						
	OUT	GPO_PF6	01		UPC2:RxAddr[1] UTOPIA master UPC2:RADR[1] POS master	01		UCC2:TxD[1]/TCG[1] Enet UCC2:TxD[1] SER	01		$\overline{\text{PCI\_RESET\_OUT}}$	01	
PF7	IN	GPI_PF7	10								PCI_C/$\overline{\text{BE}}$[0]	11	GND
	OUT	GPO_PF7	01		UPC2:TxData[3] UTOPIA 16[1] UPC2:TDAT[3] POS	01							
PF8	IN	GPI_PF8	10								PCI_C/$\overline{\text{BE}}$[1]	11	GND
	OUT	GPO_PF8	01		UPC2:TxData[2] UTOPIA 16[1] UPC2:TDAT[2] POS	01							
PF9	IN	GPI_PF9	10								PCI_C/$\overline{\text{BE}}$[2]	11	GND
	OUT	GPO_PF9	01		UPC2:TxData[1] UTOPIA 16[1] UPC2:TDAT[1] POS	01							
PF10	IN	GPI_PF10	10								PCI_C/$\overline{\text{BE}}$[3]	11	GND
	OUT	GPO_PF10	01		UPC2:TxData[0] UTOPIA 16[1] UPC2:TDAT[0] POS	01							
PF11	IN	GPI_PF11	10		UPC2:RxSOC UTOPIA master UPC2:TxSOC UTOPIA slave UPC2:RSOP POS master UPC2:TSOP POS slave (Primary Option)	10	PE25				PCI_PAR	11	GND
	OUT	GPO_PF11	01										
PF12	IN	GPI_PF12	10		UPC2:$\overline{\text{TxEnb}}$ UTOPIA slave UPC2:TENB POS slave (Primary Option)	10	PE26				$\overline{\text{PCI\_FRAME}}$	11	VCC
	OUT	GPO_PF12	01		UPC2:$\overline{\text{RxEnb}}$[0] UTOPIA master UPC2:RENB[0] POS master	01							

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

Pin	Pin Functions												
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF13	IN	GPI_PF13	10		UPC2:RxClav[0] UTOPIA master UPC2:PRPA[0]/ DRPA[0] POS master (Primary Option)	10	PE27				$\overline{\text{PCI\_TRDY}}$	11	VCC
	OUT	GPO_PF13	01		UPC2:TxClav UTOPIA slave UPC2:PTPA/ DTPA POS slave	01							
PF14	IN	GPI_PF14	10		UPC2:RxData[7] UTOPIA 8[1] UPC2:RxData[15] UTOPIA 16[1] UPC2:RDAT[15] POS (Primary Option)	10	PB2				$\overline{\text{PCI\_IRDY}}$	11	VCC
	OUT	GPO_PF14	01										
PF15	IN	GPI_PF15	10		UPC2:RxData[6] UTOPIA 8[1] UPC2:RxData[14] UTOPIA 16[1] UPC2:RDAT[14] POS (Primary Option)	10	PB3				$\overline{\text{PCI\_STOP}}$	11	VCC
	OUT	GPO_PF15	01										
PF16	IN	GPI_PF16	10		UPC2:RxData[5] UTOPIA 8[1] UPC2:RxData[13] UTOPIA 16[1] UPC2:RDAT[13] POS (Primary Option)	10	PC25				$\overline{\text{PCI\_DEVSEL}}$	11	VCC
	OUT	GPO_PF16	01										
PF17	IN	GPI_PF17	10		UPC2:RxData[4] UTOPIA 8[1] UPC2:RxData[12] UTOPIA 16[1] UPC2:RDAT[12] POS (Primary Option)	10	PB9				PCI_IDSEL	10	GND
	OUT	GPO_PF17	01										

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

Pin	Pin Functions												
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF18	IN	GPI_PF18	10		UPC2:RxData[3] UTOPIA 8[1] UPC2:RxData[11] UTOPIA 16[1] UPC2:RDAT[11] POS (Primary Option)	10	PB10				$\overline{\text{PCI\_SERR}}$	11	VCC
	OUT	GPO_PF18	01										
PF19	IN	GPI_PF19	10		UPC2:RxData[2] UTOPIA 8[1] UPC2:RxData[10] UTOPIA 16[1] UPC2:RDAT[10] POS (Primary Option)	10	PE19				$\overline{\text{PCI\_PERR}}$	11	VCC
	OUT	GPO_PF19	01										
PF20	IN	UPC2:STPA POS master (Primary Option)	10	PA20				UCC2:Gigabit RX_CLK Enet (Primary Option)	10	PA31	$\overline{\text{PCI\_REQ}}$[0]	11	VCC
	OUT	GPO_PF20	01		UPC2:STPA POS slave	01		UPC2:$\overline{\text{TxEnb}}$[2] UTOPIA UPC2:TENB[2] POS	01				
PF21	IN	GPI_PF21	10		UPC2:TxAddr[2] UTOPIA slave UPC2:TADR[2] POS slave (Primary Option)	10	PA22	CPCI_HS_ES	10	GND	$\overline{\text{PCI\_REQ}}$[1]	10	VCC
	OUT	GPO_PF21	01		UCC2:TxD[2]/TCG[2] Enet UCC2:TxD[2] SER	01		UPC2:RxAddr[2] UTOPIA master UPC2:RADR[2] POS master	01				
PF22	IN	GPI_PF22	10		UPC2:RxAddr[3] UTOPIA slave UPC2:RADR[3] POS slave (Primary Option)	10	PA16				$\overline{\text{PCI\_REQ}}$[2]	10	VCC
	OUT	GPO_PF22	01		UPC2:TxAddr[3] UTOPIA master UPC2:TADR[3] POS master	01		UCC2:TX_EN/TCG[8] Enet UCC2:$\overline{\text{RTS}}$ SER	01				

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

Pin	Pin Functions											
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF23	IN	GPI_PF23	10		UPC2:TxAddr[3] UTOPIA slave UPC2:TADR[3] POS slave (Primary Option)	10	PA14	UCC2:RxD[0]/RCG[0] Enet UCC2:RxD[0]/ UCC2:RxD (Serial) SER (Secondary Option)	10	GND	$\overline{\text{PCI\_GNT}}$[0]	11	VCC
	OUT	GPO_PF23	01		UPC2:RxAddr[3] UTOPIA master UPC2:RADR[3] POS master	01							
PF24	IN	GPI_PF24	10		UCC2:RxD[1]/RCG[1] Enet UCC2:RxD[1] SER (Secondary Option)	10	GND	UPC2:TxAddr[4] UTOPIA slave UPC2:TADR[4] POS slave (Primary Option)	10	PC24			
	OUT	GPO_PF24	01		UPC2:RxAddr[4] UTOPIA master UPC2:RADR[4] POS master	01		CPCI_HS_LED	01		$\overline{\text{PCI\_GNT}}$[1]	01	
PF25	IN	GPI_PF25	10		UCC2:RxD[3]/RCG[3] Enet UCC2:RxD[3] SER (Secondary Option)	10	GND						
	OUT	UPC2:TEOP POS master UPC2:REOP POS slave	01		UPC2:$\overline{\text{RxEnb}}$[1] UTOPIA UPC2:RENB[1] POS	01		CPCI_HS_ENUM	01		$\overline{\text{PCI\_GNT}}$[2]	01	
PF26	IN	GPI_PF26	10					DMA_DREQ1 (Primary Option)	10	PF3			
	OUT	GPO_PF26	01		UPC2:$\overline{\text{RxEnb}}$[3] UTOPIA UPC2:RENB[3] POS	01		UCC2:GTx Clock Enet	01		PCI_CLK_OUT[0]	01	
PF27	IN	GPI_PF27	10					USB:RXD (Primary Option)	10	PB10			
	OUT	GPO_PF27	01		UPC2:$\overline{\text{TxEnb}}$[3] UTOPIA UPC2:TENB[3] POS	01		DMA_DACK1 (Primary Option)	01		PCI_CLK_OUT[1]	01	
PF28	IN	GPI_PF28	10		Timer1:$\overline{\text{TGATE3}}$ Timer2:$\overline{\text{TGATE4}}$ (Primary Option)	10	PC25				UPC2:RxClav[3] UTOPIA UPC2:PRPA[3]/ DRPA[3] POS	10	GND
	OUT	GPO_PF28	01					UPC2:CLKO	01		PCI_CLK_OUT[2]	01	

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

Pin	Pin Functions											
	Direction	CPPARFx[SELn]=00			CPPARFx[SELn]=01			CPPARFx[SELn]=10			CPPARFx[SELn]=11		
		Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input	Function	CPDIRxF[DIRn]	Default Input
PF29	IN	GPI_PF29	10		USB:RN (Primary Option)	10	PB11				UPC2:TxClav[3] UTOPIA UPC2:PTPA[3]/DTPA[3] POS	10	GND
	OUT	GPO_PF29	01					PTP_PPS3 (Secondary Option)	01		PCI_SYNC_OUT	01	

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

Table 3-18. Port G Dedicated Pin Assignment

Pin	Pin Functions											
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG0	IN	GPI_PG0	10		UPC2:RxData[0] UTOPIA 8[1] UPC2:RxData[8] UTOPIA 16[1] UPC2:RDAT[8] POS (Primary Option)	10	PE14				PCI_AD[0]	11	GND
	OUT	GPO_PG0	01										
PG1	IN	GPI_PG1	10		UPC2:RxData[4] UTOPIA 16[1] UPC2:RDAT[4] POS	10	GND				PCI_AD[1]	11	GND
	OUT	GPO_PG1	01										
PG2	IN	GPI_PG2	10		UPC2:RxData[3] UTOPIA 16[1] UPC2:RDAT[3] POS	10	GND				PCI_AD[2]	11	GND
	OUT	GPO_PG2	01										
PG3	IN	GPI_PG3	10		UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Primary Option)	10	PB4				PCI_AD[3]	11	GND
	OUT	GPO_PG3	01		UPC2:TxAddr[0] UTOPIA master UPC2:TADR[0] POS master	01							

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

Pin	Pin Functions											
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG4	IN	GPI_PG4	10					UPC2:RxAddr[1] UTOPIA slave UPC2:RADR[1] POS slave (Primary Option)	10	PD17	PCI_AD[4]	11	GND
	OUT	GPO_PG4	01					UPC2:TxAddr[1] UTOPIA master UPC2:TADR[1] POS master	01				
PG5	IN	GPI_PG5	10		USB:RP (Primary Option)	10	PB9	UPC2:RxAddr[2] UTOPIA slave UPC2:RADR[2] POS slave (Primary Option)	10	PD16	PCI_AD[5]	11	GND
	OUT	GPO_PG5	01		UPC2:TxAddr[2] UTOPIA master UPC2:TADR[2] POS master	01							
PG6	IN	GPI_PG6	10					UPC2:RxData[7] UTOPIA 16[1] UPC2:RDAT[7] POS	10	GND	PCI_AD[6]	11	GND
	OUT	GPO_PG6	01										
PG7	IN	GPI_PG7	10		UPC2:RxAddr[4] UTOPIA slave UPC2:RADR[4] POS slave (Primary Option)	10	PE20				PCI_AD[7]	11	GND
	OUT	GPO_PG7	01		UPC2:TxAddr[4] UTOPIA master UPC2:TADR[4] POS master	01							
PG8	IN	GPI_PG8	10		UPC2:RxData[2] UTOPIA 16[1] UPC2:RDAT[2] POS	10	GND				PCI_AD[8]	11	GND
	OUT	GPO_PG8	01										
PG9	IN	GPI_PG9	10								PCI_AD[9]	11	GND
	OUT	GPO_PG9	01		UPC2:TxData[3] UTOPIA 8[1] UPC2:TxData[11] UTOPIA 16[1] UPC2:TDAT[11] POS	01							
PG10	IN	GPI_PG10	10								PCI_AD[10]	11	GND
	OUT	GPO_PG10	01		UPC2:TxData[5] UTOPIA 16[1] UPC2:TDAT[5] POS	01							

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

Pin	Direction	Pin Functions											
		CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG11	IN	GPI_PG11	10		UPC2:RxClav[1] UTOPIA UPC2:PRPA[1]/ DRPA[1] POS	10	GND	UPC2:RMOD POS master UPC2:TMOD POS slave (Primary Option)	10	PC10	PCI_AD[11]	11	GND
	OUT	USB:$\overline{OE}$	01										
PG12	IN	GPI_PG12	10		UPC2:TxClav[2] UTOPIA UPC2:PTPA[2]/DTPA[2] POS	10	GND	UPC2:RVAL POS master (Primary Option)	10	PA31	PCI_AD[12]	11	GND
	OUT	USB:TP	01		UPC2:RVAL POS slave	01							
PG13	IN	GPI_PG13	10		UPC2:TxClav[1] UTOPIA UPC2:PTPA[1]/DTPA[1] POS	10	GND				PCI_AD[13]	11	GND
	OUT	GPO_PG13	01		UPC2:TERR POS master UPC2:RERR POS slave	01		USB:TN	01				
PG14	IN	GPI_PG14	10								PCI_AD[14]	11	GND
	OUT	GPO_PG14	01		UPC2:TxData[4] UTOPIA 16[1] UPC2:TDAT[4] POS	01							
PG15	IN	GPI_PG15	10					UPC2:RxPrty UTOPIA master UPC2:TxPrty UTOPIA slave UPC2:RPRTY POS master UPC2:TPRTY POS slave (Primary Option)	10	PB12	PCI_AD[15]	11	GND
	OUT	GPO_PG15	01										
PG16	IN	GPI_PG16	10								PCI_AD[16]	11	GND
	OUT	GPO_PG16	01		UPC2:TxPrty UTOPIA master UPC2:RxPrty UTOPIA slave UPC2:TPRTY POS master UPC2:RPRTY POS slave	01							

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

Pin	Direction	Pin Functions											
		CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG17	IN	GPI_PG17	10								PCI_AD[17]	11	GND
	OUT	GPO_PG17	01		UPC2:TxSOC UTOPIA master UPC2:RxSOC UTOPIA slave UPC2:TSOP POS master UPC2:RSOP POS slave	01							
PG18	IN	GPI_PG18	10					UPC2:$\overline{\text{RxEnb}}$ UTOPIA slave UPC2:RENB POS slave (Primary Option)	10	PE15	PCI_AD[18]	11	GND
	OUT	GPO_PG18	01					UPC2:$\overline{\text{TxEnb}}$[0] UTOPIA master UPC2:TENB[0] POS master	01				
PG19	IN	GPI_PG19	10		UPC2:TxClav[0] UTOPIA master UPC2:PTPA[0]/DTPA[0] POS master (Primary Option)	10	PE16				PCI_AD[19]	11	GND
	OUT	GPO_PG19	01		UPC2:RxClav UTOPIA slave UPC2:PRPA/DRPA POS slave	01							
PG20	IN	GPI_PG20	10			11	GND				PCI_AD[20]	11	GND
	OUT	GPO_PG20	01					UPC2:TxData[7] UTOPIA 8[1] UPC2:TxData[15] UTOPIA 16[1] UPC2:TDAT[15] POS	01				
PG21	IN	GPI_PG21	10			11	GND				PCI_AD[21]	11	GND
	OUT	GPO_PG21	01					UPC2:TxData[6] UTOPIA 8[1] UPC2:TxData[14] UTOPIA 16[1] UPC2:TDAT[14] POS	01				
PG22	IN	GPI_PG22	10			11	GND				PCI_AD[22]	11	GND
	OUT	GPO_PG22	01					UPC2:TxData[5] UTOPIA 8[1] UPC2:TxData[13] UTOPIA 16[1] UPC2:TDAT[13] POS	01				

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

Pin	Pin Functions												
	Direction	CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG23	IN	GPI_PG23	10			11	GND				PCI_AD[23]	11	GND
	OUT	GPO_PG23	01					UPC2:TxData[4] UTOPIA 8[1] UPC2:TxData[12] UTOPIA 16[1] UPC2:TDAT[12] POS	01				
PG24	IN	GPI_PG24	10		UPC2:RxClav[2] UTOPIA UPC2:PRPA[2]/ DRPA[2] POS	10	GND	UPC2:RERR POS master UPC2:TERR POS slave (Primary Option)	10	PA19	PCI_AD[24]	11	GND
	OUT	GPO_PG24	01					UCC2:TxD[3]/TCG[3] Enet UCC2:TxD[3] SER	01				
PG25	IN	GPI_PG25	10								PCI_AD[25]	11	GND
	OUT	GPO_PG25	01		UPC2:TxData[2] UTOPIA 8[1] UPC2:TxData[10] UTOPIA 16[1] UPC2:TDAT[10] POS	01							
PG26	IN	GPI_PG26	10								PCI_AD[26]	11	GND
	OUT	GPO_PG26	01		UPC2:TxData[1] UTOPIA 8[1] UPC2:TxData[9] UTOPIA 16[1] UPC2:TDAT[9] POS	01							
PG27	IN	GPI_PG27	10								PCI_AD[27]	11	GND
	OUT	GPO_PG27	01		UPC2:TxData[0] UTOPIA 8[1] UPC2:TxData[8] UTOPIA 16[1] UPC2:TDAT[8] POS	01							
PG28	IN	GPI_PG28	10								PCI_AD[28]	11	GND
	OUT	GPO_PG28	01		UPC2:TxData[7] UTOPIA 16[1] UPC2:TDAT[7] POS	01							
PG29	IN	GPI_PG29	10								PCI_AD[29]	11	GND
	OUT	GPO_PG29	01		UPC2:TxData[6] UTOPIA 16[1] UPC2:TDAT[6] POS	01							

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

Pin	Direction	Pin Functions											
		CPPARGx[SELn]=00			CPPARGx[SELn]=01			CPPARGx[SELn]=10			CPPARGx[SELn]=11		
		Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input	Function	CPDIRxG[DIRn]	Default Input
PG30	IN	GPI_PG30	10		UPC2:REOP POS master UPC2:TEOP POS slave (Primary Option)	10	PC14	UCC2:RxD[2]/RCG[2] Enet UCC2:RxD[2] SER (Secondary Option)	10	GND	PCI_AD[30]	11	GND
	OUT	GPO_PG30	01		UPC2:$\overline{\text{TxEnb}}$[1] UTOPIA UPC2:TENB[1] POS	01							
PG31	IN	GPI_PG31	10					UCC2:RX_DV/RCG[8] Enet UCC2:$\overline{\text{CTS}}$ SER (Secondary Option)	10	GND	PCI_AD[31]	11	GND
	OUT	GPO_PG31	01		UPC2:$\overline{\text{RxEnb}}$[2] UTOPIA UPC2:RENB[2] POS	01		UPC2:TMOD POS master UPC2:RMOD POS slave	01				

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

Part II
Reset and Configuration

Part II includes the following chapters:

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

Chapter 4
Reset, Clocking, and Initialization

The reset, clocking, and control signals offer many options for the 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 located on device pins that have other functions when the device is not in reset. These input signals 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 input 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 located on a device pin that has another function when the device is not in reset state. This signal is sampled during the assertion of 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	Input during power-on and hard reset flows. Functional signal after reset flow completes.
PCI_MODE	I	A system configuration signal, but its state is sampled during the assertion of PORESET and affects the default values of registers. For a description of this signal, see Table 5-31 and Table 5-32 in Chapter 5, “System Configuration”.	

4.1.2 Clock Signals

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

Table 4-2. External Clock Signals

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

4.2 Functional Description

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

4.2.1 Reset Operations

The device has several inputs to the reset logic:

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

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

4.2.1.2 Reset Actions

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

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

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

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

Table 4-4. Reset Actions

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

4.2.2 Power-On Reset Flow

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

Directly after the negation of $\overline{\text{PORESET}}$, the device starts the configuration process. The device asserts $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the power-on reset process, including configuration. Configuration time varies according to the configuration source and CLKIN (PCI host mode) or PCI_CLK (PCI agent

mode) frequency. Initially, the reset configuration inputs are sampled to determine the configuration source and the input clock division mode. Next, the device starts loading the reset configuration words. The system PLL begins to lock according to the clock mode values in the reset configuration word low. When the system PLL is locked, the clock unit starts distributing clock signals in the device. At this stage, the core PLL begins to lock. When it is locked and the reset configuration words are loaded, $\overline{\text{HRESET}}$ is released; $\overline{\text{SRESET}}$ is released 16 clocks later.

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

1. Power is applied to meet the specifications in the device data sheet.
2. The system asserts $\overline{\text{PORESET}}$ and $\overline{\text{TRST}}$, causing all registers to be initialized to their default states and most I/O drivers to be released to high-impedance (some clock, clock enabled, and system control signals remain active).
3. The system applies a stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) signal and stable reset configuration inputs (CFG_RESET_SOURCE, CFG_CLKIN_DIV).
4. The system negates $\overline{\text{PORESET}}$ after at least 32 stable CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) clock cycles.
5. The device samples the reset configuration input signals to determine the clock division and the reset configuration words source.
6. The device starts loading the reset configuration words. Loading time depends on the reset configuration word source.
7. When the reset configuration word low is loaded, the system PLL and QUICC Engine PLL begin to lock. When the system PLL is locked, *csb_clk* is supplied to the core PLL.
8. The core PLL begins to lock.
9. The device drives $\overline{\text{HRESET}}$ asserted until the e300 PLL is locked and the reset configuration words are loaded.
10. The user optionally negates $\overline{\text{HRESET}}$ if it was not negated earlier.

NOTE:

JTAG logic must always be initialized by asserting $\overline{\text{TRST}}$. If the JTAG signals are not used, $\overline{\text{TRST}}$ should be connected directly to $\overline{\text{PORESET}}$. $\overline{\text{TRST}}$ must not remain asserted after the negation of $\overline{\text{PORESET}}$. There is no need to assert the $\overline{\text{SRESET}}$ signal when $\overline{\text{HRESET}}$ is asserted.

11. The internal reset to the core and the rest of the logic is negated. I/O drivers are enabled. The LBC DLL begin to lock. The PCI interface can assert $\overline{\text{DEVSEL}}$ in response to configuration cycles.
12. The device stops driving $\overline{\text{SRESET}}$ and $\overline{\text{SRESET}}$ is negated. The reset to the e300 core is negated and the core is enabled.The boot sequencer, if enabled, is released, causing it to load configuration data from serial ROMs, as described in Section 15.4.5, “Boot Sequencer Mode.”
13. Before the boot sequencer finishes, it can enable the PCI interface to accept external requests, if required, by clearing the CFG_LOCK bit in the PCI function configuration register as described in Table 13-40. If the e300 core is required to proceed, the boot sequencer should enable boot vector fetch by clearing ACR[COREDIS] as described in Section 6.2.1, “Arbiter Configuration Register (ACR).”

14. The PCI interface can now accept external requests, if enabled, and the boot vector fetch by the core can proceed, if enabled. The device is now in its ready state.

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

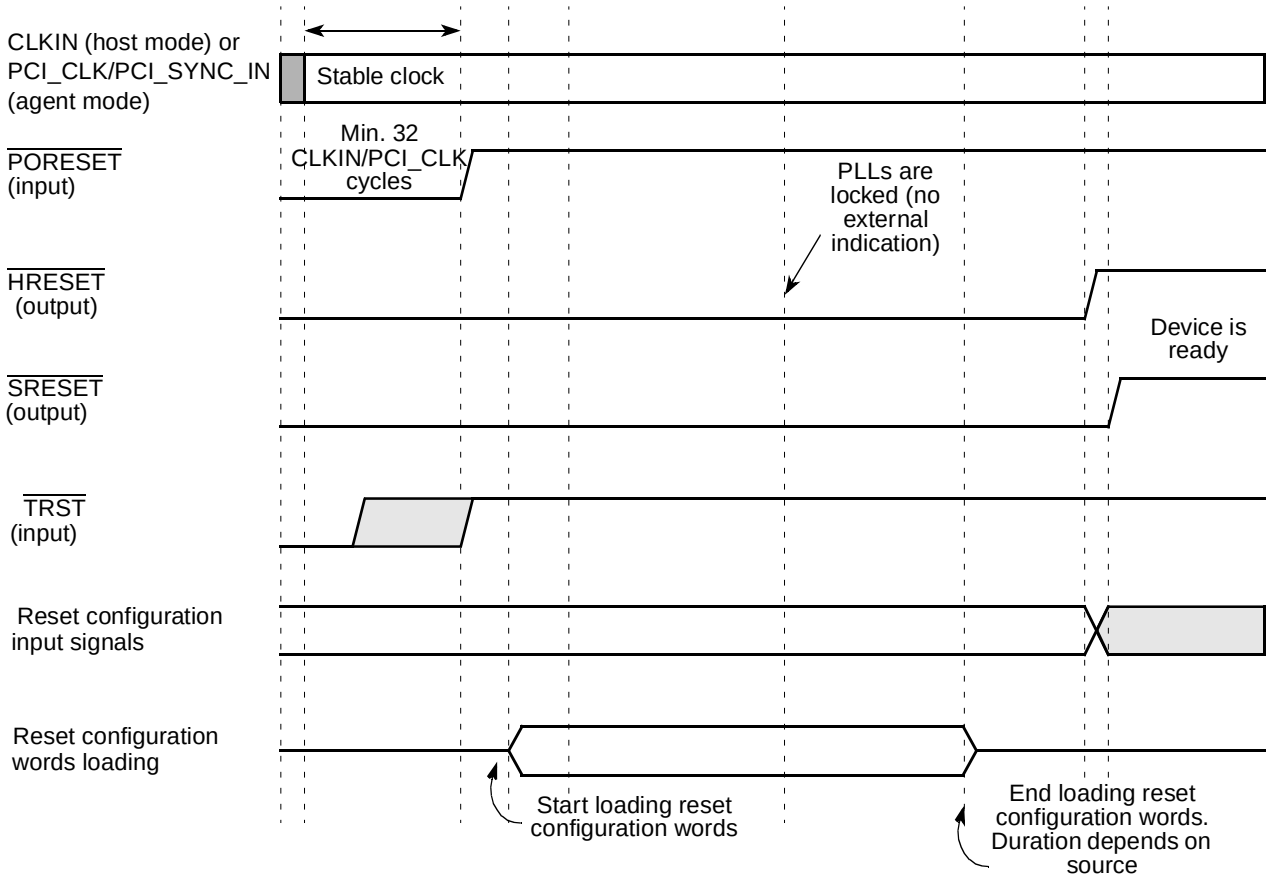


Figure 4-1. Power-On Reset Flow

4.2.3 Hard Reset Flow

The $\overline{\text{HRESET}}$ signal is initiated externally by asserting $\overline{\text{HRESET}}$ or internally when the device detects a reason to generate an internal hard reset sequence. In both cases the device continues asserting $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ throughout the $\overline{\text{HRESET}}$ state. The hard reset sequence time varies according to the configuration source and CLKIN (PCI host mode) or PCI_CLK (PCI agent mode) frequency. The reset configuration input signals (CFG_RESET_SOURCE and CFG_CLKIN_DIV) are not sampled by hard reset (only by power-on reset), so the device immediately starts loading the reset configuration words, and configures the device as explained in Section 4.3.3, "Loading the Reset Configuration Words." After the configuration sequence completes, the device releases both the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ signals and exits the $\overline{\text{HRESET}}$ state. An external pull-up resistor should negate the signals. After negation is detected, a 16-cycle period is taken before testing for the presence of an external (hard/soft) reset.

NOTE

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

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

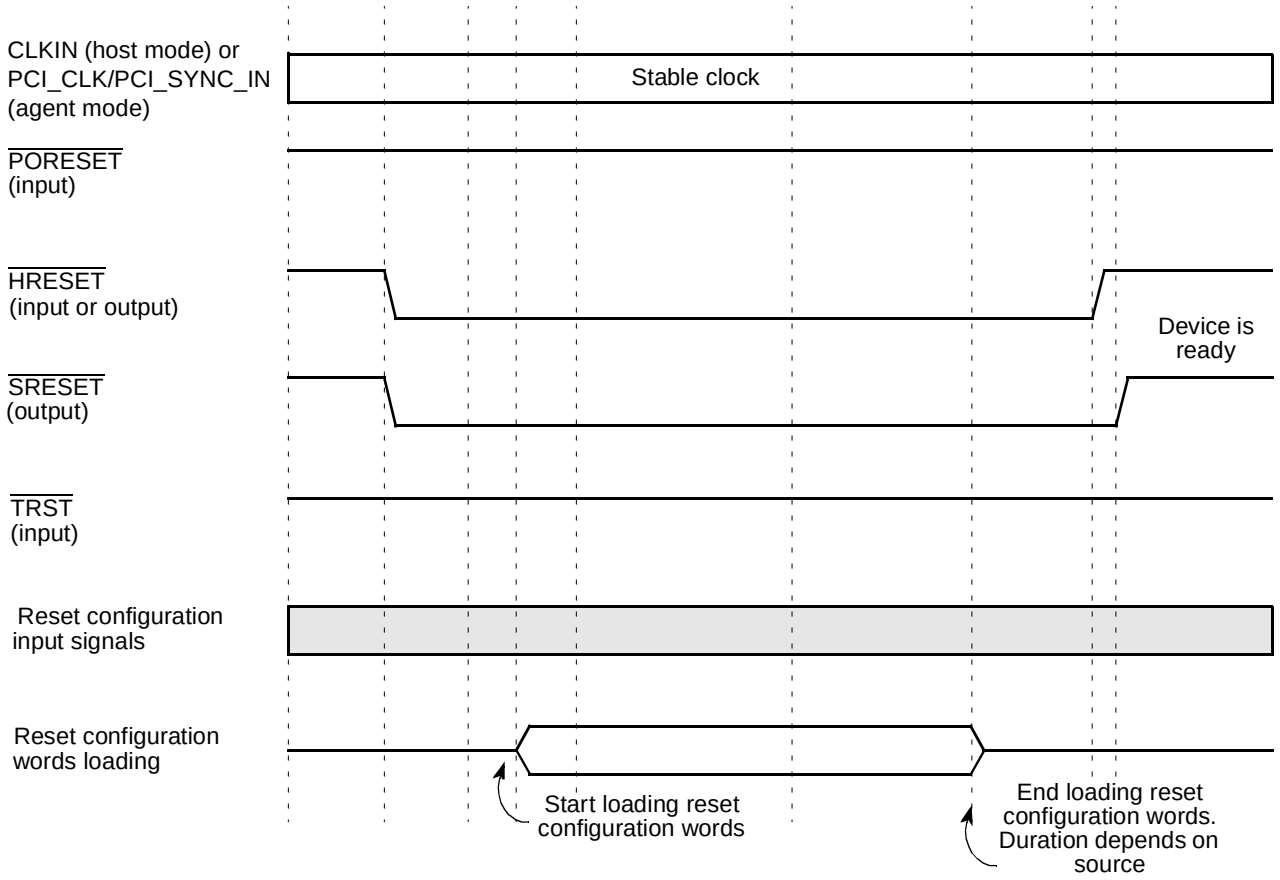


Figure 4-2. Hard Reset Flow

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, after which the device 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 located 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. During the period in which 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 (I^2C #1) or uses hard-coded default options. The value of these signals also affects the duration of power-on and hard reset sequences. In any case, the reset sequence does not exceed 1 ms.

Table 4-5. Reset Configuration Words Source

CFG_RESET_SOURCE[0:2]	Meaning
000	Reset configuration word is loaded from a local bus EEPROM.
001	Reset configuration word is loaded from an I^2C EEPROM. PCI_CLK/PCI_SYNC_IN is in the range of 25–44 MHz. This option will be removed from future designs. Thus, customers are recommended to use 010 option.

Table 4-5. Reset Configuration Words Source (continued)

CFG_RESET_SOURCE[0:2]	Meaning
010	Reset configuration word is loaded from an I^2C EEPROM. PCI_CLK/PCI_SYNC_IN 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 CFG_CLKIN_DIV configuration input selects the relationship between CLKIN and PCI_SYNC_OUT/SYNC_OUT as shown in Table 4-7. As a PCI host, the device supports three PCI_CLK output signals. The frequency of the output clocks will be equal to the PCI_SYNC_OUT frequency.

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

Table 4-6. CLKIN Division

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

4.3.1.3 Selecting Reset Configuration Input Signals

The example described in Table 4-7 shows how the user should pull down or pull up the reset configuration input signals (CFG_RESET_SOURCE, CFG_CLKIN_DIV). The reset sequence duration is measured from the negation of $\overline{\text{PORESET}}$ 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)	CFG_CLKIN_DIV (Host Mode)	PCI_CLK Frequency (Agent Mode)	CFG_RESET_ SOURCE[0:2]	Reset Sequence Duration in CLKIN/ PCI_CLK Cycles	Duration
No	33 MHz	0	33 MHz	000, 011–111 (not I^2C EEPROM)	15380	462 µs
No	66 MHz	0	66 MHz	000, 011–111 (not I^2C EEPROM)	15380	231 µs
No	66 MHz	1	33 MHz	000, 011–111 (not I^2C EEPROM)	30760/15380	462 µs
Yes	33 MHz	0	33 MHz	001 (I^2C EEPROM, low PCI_SYNC_IN/ PCI_CLK clock frequency)	24548	736 µs
Yes	66 MHz	0	66 MHz	010 (I^2C EEPROM, high PCI_SYNC_IN/ PCI_CLK clock frequency)	37908	568 µs
Yes	66 MHz	1	33 MHz	001 (I^2C EEPROM, low PCI_SYNC_IN/ PCI_CLK clock frequency)	49096/24548	736 µs

4.3.2 Reset Configuration Words

The reset configuration words control the clock ratios and other basic device functions such as PCI host or agent mode, boot location, and endian mode. The reset configuration words are loaded from the local bus or from the 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 value 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. Modifying values in these other units' memory mapped registers do 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)

The register map for these registers is described in Section 4.5, "Memory Map/Register Definition."

4.3.2.1 Reset Configuration Word Low Register (RCWLR)

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

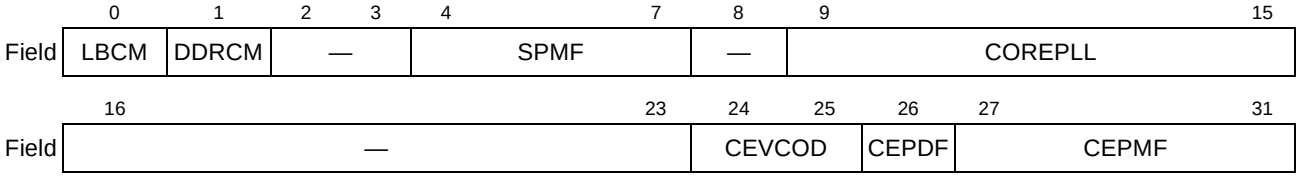


Figure 4-3. Reset Configuration Word Low Register (RCWLR)

Table 4-8 defines the reset configuration word low bit fields.

Table 4-8. Reset Configuration Word Low Bit Settings

Bits	Name	Description
0	LBCM	Local bus/secondary DDR SDRAM memory controller clock mode. Selects the local bus and secondary DDR memory controller clock ratio. If this bit is set, the local bus and secondary DDR memory controllers operate at twice the frequency of the *csb_clk*. If this bit is cleared, the local bus and secondary DDR memory controllers operate at the *csb_clk* frequency. The 2:1 mode is useful when the csb_clk operates at low frequency. 0 *csb_clk* ratio is 1:1 1 *csb_clk* ratio is 2:1
1	DDRCM	DDR SDRAM memory controller clock mode. Selects the DDR SDRAM memory controller clock ratio. If this bit is set, the DDR SDRAM memory controller operates at twice the frequency of the *csb_clk*. If this bit is cleared, the DDR SDRAM memory controller operates at the *csb_clk* frequency. The 2:1 mode is useful mostly for a 32-bit data bus width. 0 *csb_clk* ratio is 1:1 1 *csb_clk* ratio is 2:1
2–3	—	Reserved, should be cleared.
4–7	SPMF	System PLL multiplication factor. See Section 4.3.2.1.1, "System PLL Configuration," for more information.
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.

Table 4-8. Reset Configuration Word Low Bit Settings (continued)

Bits	Name	Description
24–25	CEVCOD	QUICC Engine PLL VCO division. Establishes the internal ratio between the QUICC Engine PLL VCO point and the QUICC Engine PLL output. 00 4 01 8 10 2 11 Reserved, should be cleared **Notes**:Set CEVCOD to 01 (Division factor of 8) for QE frequency of below 100 MHz. Set CEVCOD to 00 (Division factor of 4) for QE frequency of 100–267 MHz. Set CEVCOD to 10 (Division factor of 2) for QE frequency of above 267 MHz.
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.

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_cfg_clkin_div))(PCI agent mode)**
4–7	SPMF	0000	16 : 1
		0001	Reserved
		0010	2 : 1
		0011	3 : 1
		0100	4 : 1
		0101	5 : 1
		0110	6 : 1
		0111	7 : 1
		1000	8 : 1
		1001	9 : 1
		1010	10 : 1
		1011	11 : 1
		1100	12 : 1
		1101	13 : 1
		1110	14 : 1
		1111	15: 1

NOTE

In PCI host mode, the SPMF field described in Table 4-9 always selects the *csb_clk*:CLKIN ratio regardless of the CFG_CLKIN_DIV reset configuration input value during reset flow.

The SPMF field maximum allowed value is dependent on the value sampled on CFG_CLKIN_DIV during power-on reset, and the LBCM and DDRCM reset configuration word fields values. Table 4-10 defines the upper limit of SPMF with respect to these values. Values for SPMF are as follows:

Table 4-10. SPMF Maximum Values

CFG_CLKIN_DIV	LBCM	DDRCM	Maximum SPMF Value (decimal)
0	0	0	16
0	0	1	8
0	1	0	8
0	1	1	8
1	0	0	8
1	0	1	4
1	1	0	4
1	1	1	4

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,
		00001	Reserved
		00010	2
		00011	3
		00100	4
		00101	5
		00110	6
		00111	7
		01000	8
		01001	9
		01010	10
		01011	11
		01100	12
		01101	13
		01110	14
		01111	15
		10000	16
		10001	17
		10010	18
		10011	19
		10100	20
		10101	21
		10110	22
		10111	23
		11000	24
		11001	25
		11010	26
		11011	27
		11100	28
		11101	29
		11110	30
		11111	31

4.3.2.2 Reset Configuration Word High Register (RCWHR)

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

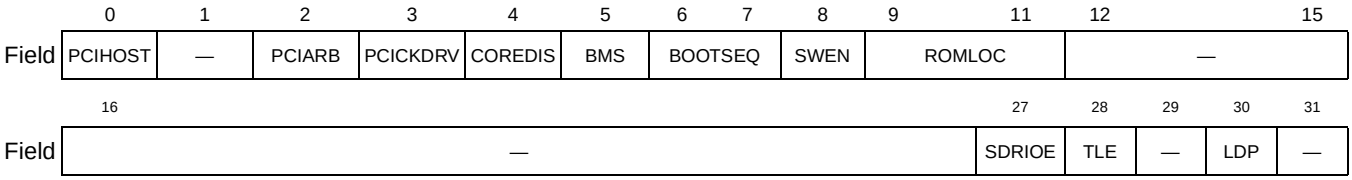


Figure 4-4. Reset Configuration Word High Register (RCWHR)

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.
2	PCIARB	PCI internal arbiter mode. Enables the on-chip PCI arbiter. 0 On-chip PCI arbiter is disabled. External arbitration is required. 1 On-chip PCI arbiter is enabled. **Pin Function When PCIARB = 0** / **Pin Function When PCIARB = 1** CPCI_HS_ES / $\overline{\text{PCI\_REQ}}$[1] CPCI_HS_LED / $\overline{\text{PCI\_GNT}}$[1] CPCI_HS_ENUM / $\overline{\text{PCI\_GNT}}$[2]
3	PCICKDRV	PCI clock output drive. Enables the output buffers of the PCI clock output signals (PCI_CLK_OUT[0:2]) when the MPC8360E is in host mode. Disabling the PCI clock output buffers allows for external PCI clock distribution and alternate QUICC Engine functionality on the PCI_CLK_OUT[0:2] signals. In host mode when PCICKDRV = 0, the PCI_SYNC_IN signal becomes the primary clock input for the device. 0 Buffers are disabled. 1 Buffers are enabled.
4	COREDIS	Core disable mode. Specifies the e300 core mode out of reset. If COREDIS is set, the core cannot fetch boot code until it is configured by an external master. The external master frees the core to boot by clearing the COREDIS bit in the arbiter configuration register as described in Section 6.2.1, "Arbiter Configuration Register (ACR)." This bit must be set when the boot sequencer is enabled to initiate the device (BOOTSEQ is not 0b00). Otherwise, unpredictable behavior occurs. 0 The core can boot without waiting for configuration by an external master. 1 Core boot holdoff mode. The core is prevented from booting until it is configured by an external master.
5	BMS	Boot memory space. See Section 4.3.2.2.2, "Boot Memory Space (BMS)," for more information.

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

Bits	Name	Description
6–7	BOOTSEQ	Boot sequencer configuration. See Section 4.3.2.2.3, "Boot Sequencer Configuration," for more information.
8	SWEN	Software watchdog enable. Selects whether the software watchdog is enabled to start counting down immediately when coming out of reset. The user can override this value by writing to the system watchdog control register (SWCRR[SWEN]) during system initialization. 0 Disabled. 1 Enabled.
9–11	ROMLOC	Boot ROM interface location. See Section 4.3.2.2.4, "Boot ROM Location," for more information.
12–15	—	Reserved, should be cleared.
16–17	—	Reserved, should be cleared.
18–19	—	Reserved, should be cleared.
20–26	—	Reserved, should be cleared.
27	SDDRIOE	Secondary DDR IO enable. See Section 4.3.2.2.5, "Secondary DDR IO Enable," for more information.
28	TLE	True little-endian. See Section 4.3.2.2.6, "e300 Core True Little-Endian," for more information.
29	—	Reserved, should be cleared.
30	LDP	LDP pin mux state after reset. See Section 4.3.2.2.7, "LDP Configuration," for more information.
31	—	Reserved, should be cleared.

4.3.2.2.1 PCI Host/Agent Configuration

The PCIHOST configuration parameter, shown in Table 4-13, 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

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

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

NOTE

In PCI host mode, although selecting the PCI option with ROMLOC sets the appropriate local access window, $\overline{\text{PCI\_RESET\_OUT}}$ remains asserted and PCI_CLK_OUT[x] are disabled after reset.

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

4.3.2.2.5 Secondary DDR IO Enable

The SDDRIOE bit reset configuration word field, shown in Table 4-17, enables the Secondary DDR memory controller I/O pins.

Table 4-17. Secondary DDR Configuration

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

4.3.2.2.6 e300 Core True Little-Endian

The true little endian reset configuration word field, shown in Table 4-18, selects whether the e300 core operates in big-endian mode or true little-endian mode at reset.

Table 4-18. 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

Table 4-19. LALE Configuration

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

4.3.2.2.7 LDP Configuration

The LDP reset configuration word field configures the initial state of SICRL[LDP_A], which controls the functionality of the LPD[0:3] pins. Table 4-20 shows the LDP configuration.

Table 4-20. LDP Configuration

Reset Configuration Word High Register (RCWHR) Bit	Field Name	Meaning
30	LDP	0 Initial value of SICRL[LDP_A] is 1, meaning that LDP0–LDP3 are used for local data parity. 1 Initial value of SICRL[LDP_A] is 0; the meaning is as follows: • LDP0 is used as $\overline{\text{CKSTOP\_OUT}}$ • LDP1 is used as $\overline{\text{CKSTOP\_IN}}$. • LDP2 is used as LCS6 • LDP3 is used as LCS7

4.3.3 Loading the Reset Configuration Words

The device loads the reset configuration words from a local bus EEPROM, or an 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 in detail.

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 4-21 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 4-21 are always read on byte lane LAD[0:7] regardless of the port size.

Table 4-21. 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 are used to 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 4-22 shows the data structure of the local bus device containing the reset configuration words (RCWL and RCWH).

Table 4-22. 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				

4.3.3.1.1 Local Bus EEPROM Timing

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).

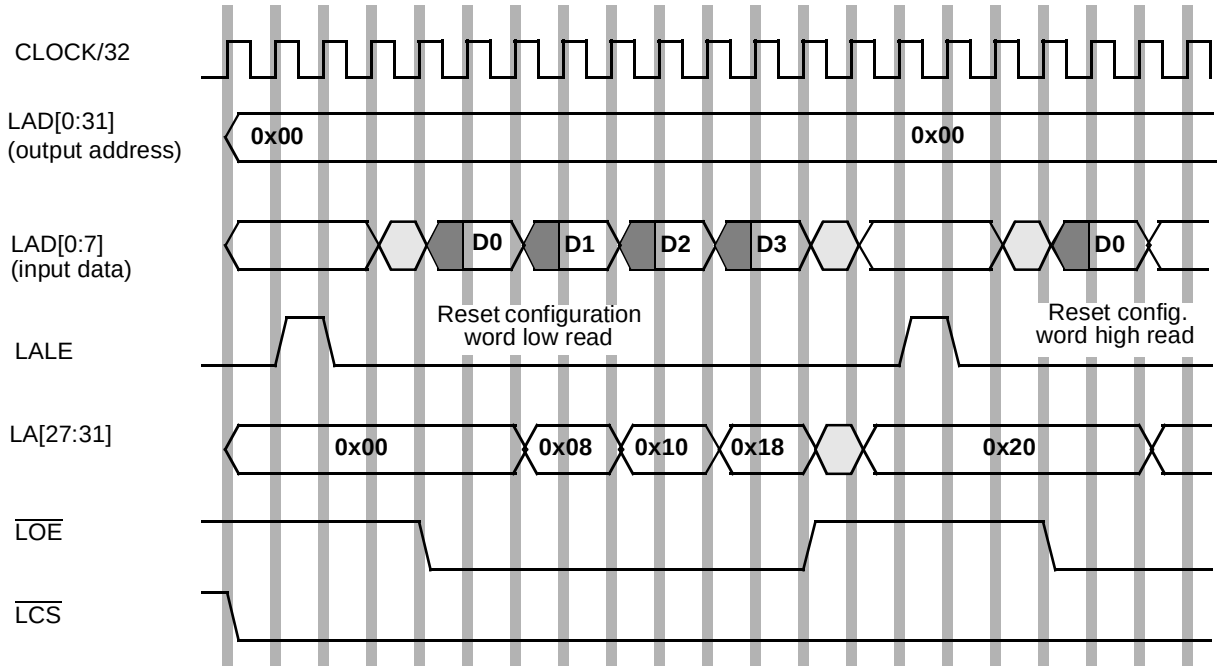


Figure 4-5. Loading Reset Configuration Words from Local Bus

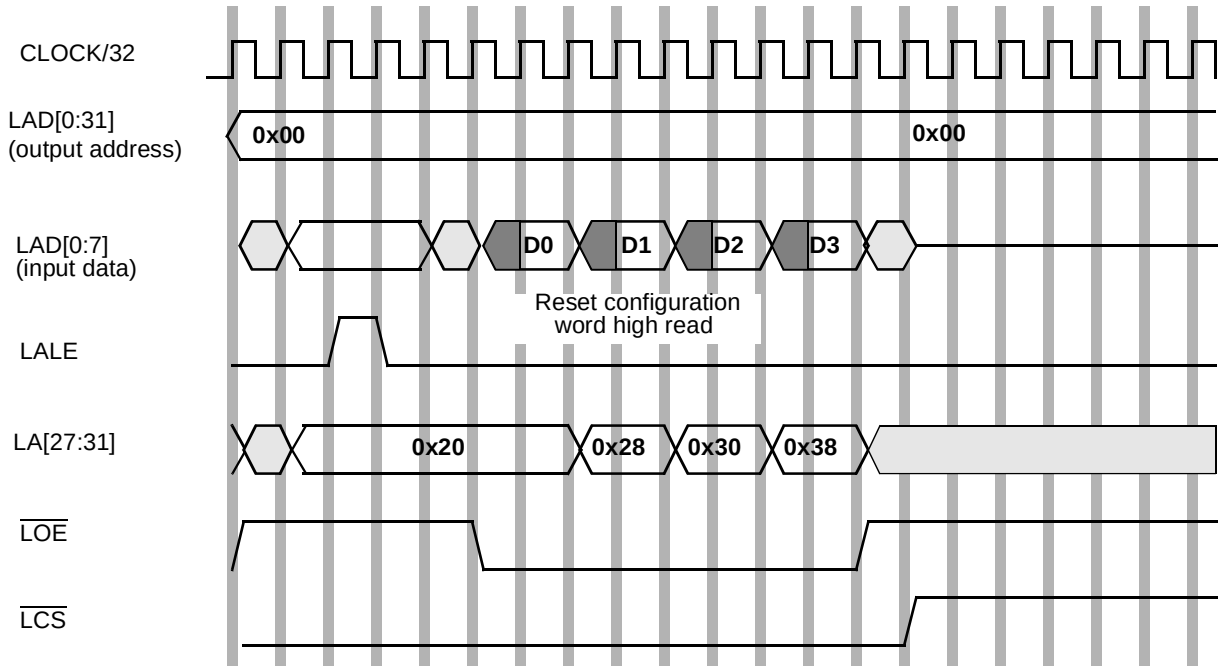


Figure 4-6. Loading Reset Configuration Words from Local Bus (continued)

4.3.3.2 Loading from I²C EEPROM

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

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

4.3.3.2.1 Using the Boot Sequencer Reset Configuration

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

NOTE

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

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

After $\overline{\text{HRESET}}$ is negated, the functional boot sequencer, in extended I^2C 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 4-23 and Table 4-24.

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."

4.3.3.3.1 Hard-Coded Reset Configuration Word Low

Table 4-23 defines the hard-coded reset configuration word low fields values, according to the CFG_RESET_SOURCE[0:2].

Table 4-23. Hard-Coded Reset Configuration Word Low Fields Values

Bits	Name	CFG_RESET_SOURCE Value					Meaning
		011	100	101	110	111	
0	LBCM	0	0	1	1	0	LBC controller clock: *csb_clk* 0 1:1 1 2:1
1	DDRCM	0	0	1	1	0	DDR controller clock: *csb_clk* 0 1:1 1 2:1
2–3	Res	10	10	10	10	10	—
4–7	SPMF	0100	1000	0100	0101	0100	*csb_clk*: PCI_CLK ratio SPMF:1
8	Res	0	0	0	0	0	—

Table 4-23. Hard-Coded Reset Configuration Word Low Fields Values (continued)

Bits	Name	CFG_RESET_SOURCE Value					Meaning
		011	100	101	110	111	
9–15	COREPLL	0100011	0100011	0100100	0100100	0000100	Core clock: *csb_clk* ratio
16–31	CEPMF	0x0006	0x000C	0x0008	0x000A	0x0006	QUICC Engine Clock: PCI_CLK ratio CEPMF:1—

4.3.3.3.2 Hard-Coded Reset Configuration Word High Fields Values

Table 4-24 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 4-24. Hard-Coded Reset Configuration Word High Field Values

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

4.3.3.3.3 Examples for Hard-Coded Reset Configuration Words Usage

Examples for various clock modes are listed in Table 4-25.

Table 4-25. Examples For Hard-Coded Reset Configuration Words Usage

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

4.4 Clocking

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

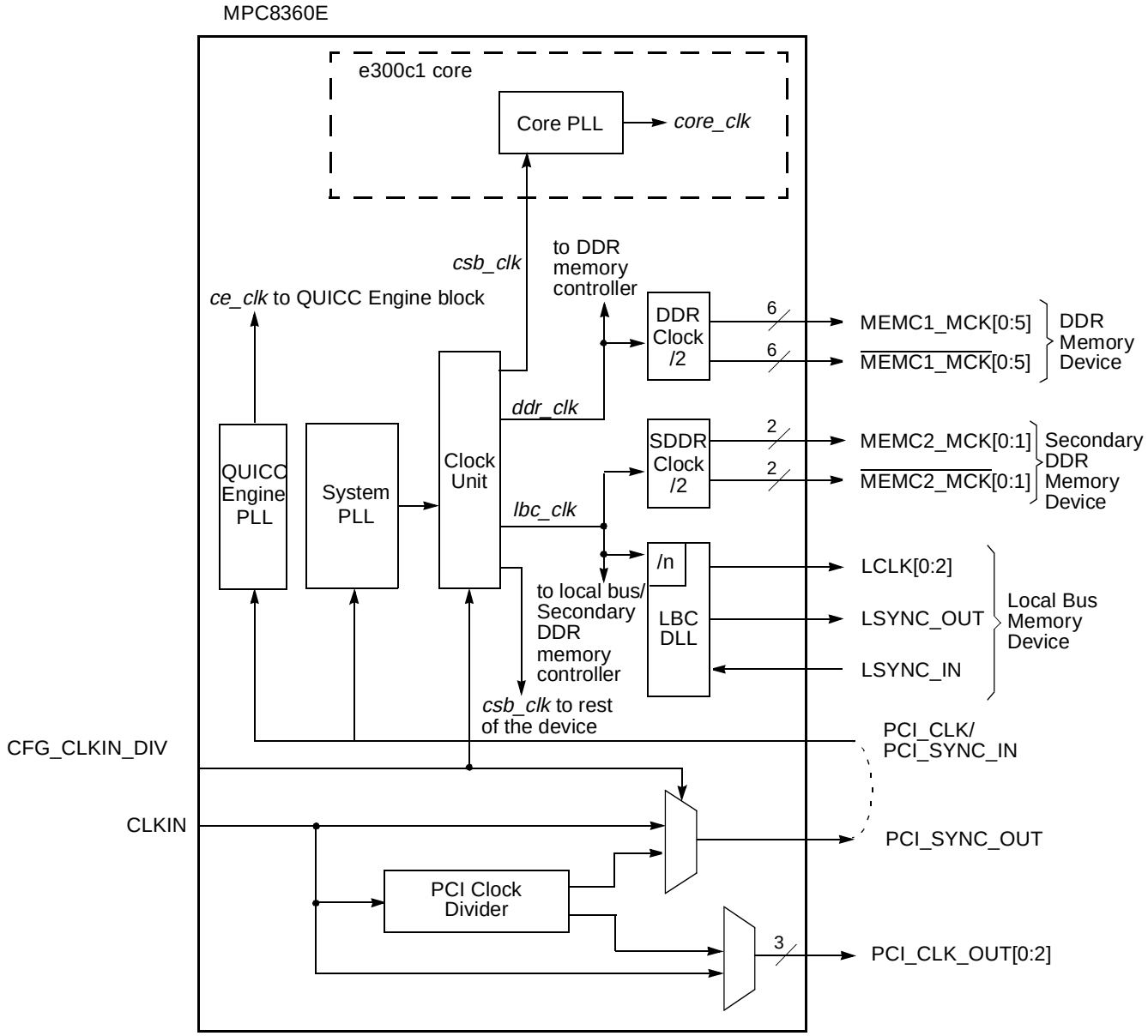


Figure 4-9. Clock Subsystem Block Diagram

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

4.4.1 Clocking in PCI Host Mode

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

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

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

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

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

The PCI clock output buffers are enabled or disabled according to RCWH[PCICKDRV]. If PCI clock output buffers are enabled, each of the individual clock outputs can be enabled (enable toggling of the clock) by setting its corresponding OCCR[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 signal should be tied to GND, and the clock output signals, PCI_CLK_OUT*n* and PCI_SYNC_OUT, are not used.

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

4.4.3 System Clock Domains

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

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

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

csb_clk = [PCI_SYNC_IN × (1 + CFG_CLKIN_DIV)] × SPMF

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

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

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

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

In addition, some of the internal units may be required to be shut off or operate at lower frequency than the *csb_clk* frequency. These units have a default clock ratio that can be configured by a memory mapped register after the device comes out of reset. Table 4-26 specifies which units have a configurable clock frequency. Refer to Section 4.5.2.3, "System Clock Control Register (SCCR)."

Table 4-26. Configurable Clock Units

Unit	Default Frequency	Options
Security core	*csb_clk*/3	Off, *csb_clk*, *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 Definition

4.5.1 Reset Configuration Registers Descriptions

The reset configuration and status registers are shown in Table 4-27.

Table 4-27. Reset Configuration and Status Registers Memory Map

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

4.5.1.1 Reset Configuration Word Low Register (RCWLR)

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

4.5.1.2 Reset Configuration Word High Register (RCWHR)

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

4.5.1.3 Reset Status Register (RSR)

The RSR shown in Figure 4-10 captures various reset events in the device.

Address 0x0_0910 Access: User read/write

	0	2	3										14	15
R W	RSTSRC		—											BSF
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 W	—		SWSR	SWHR	—		JSRS	—		CSHR	SWRS	BMRS	SRS	HRS
Reset	All zeros													

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

Figure 4-10. Reset Status Register (RSR)

Table 4-28 defines the reset status register bit fields.

Table 4-28. 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 for detailed description. 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 that a software hard reset has occurred. Cleared by writing a 1 to it (writing zero has no effect).
20–22	—	Reserved, should be cleared.
23	JSRS	JTAG soft reset status. When the JTAG reset request is set, JTRS 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 occurred 1 A JTAG reset event occurred
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 occurred 1 An enabled check stop reset event occurred

Table 4-28. 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 one to it (writing zero has no effect). 0 No software watchdog reset event occurred 1 A software watchdog reset event has occurred
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 has occurred. 1 A bus monitor reset event has occurred.
30	SRS	Soft reset status. When an external or internal soft reset event is detected, SRS is set and remains that way until software clears it. SRS is cleared by writing a 1 to it (writing zero has no effect). 0 No soft reset event has occurred. 1 A soft reset event has occurred.
31	HRS	Hard reset status. When an external or internal hard reset event is detected, HRS is set and remains that way until software clears it. HRS is cleared by writing a 1 (writing zero has no effect). 0 No hard reset event has occurred. 1 A hard reset event has occurred.

NOTE

The reset status register accumulates reset events. For example, because software watchdog expiration results in a hard reset, which in turn results in a soft reset, RSR[SWRS], RSR[SRS], and RSR[HRS] are all set after a software watchdog reset. This register returns to its reset value only when power-on reset occurs.

4.5.1.4 Reset Mode Register (RMR)

The reset mode register (RMR), shown in Figure 4-11, is used to enable a hard reset sequence on the device whenever the e300 core enters checkstop state.

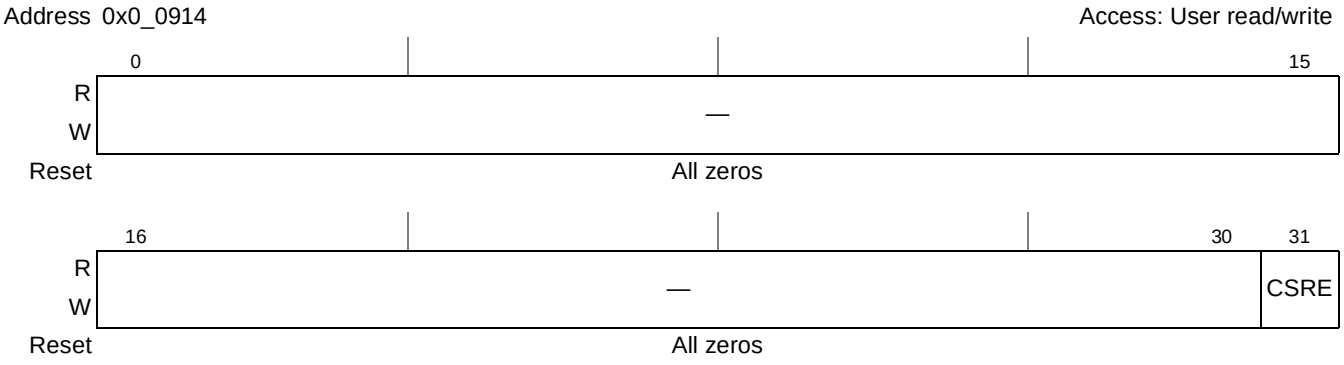


Figure 4-11. Reset Mode Register (RMR)

Table 4-29 describes RMR fields.

Table 4-29. 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 chip to perform a hard reset sequence whenever 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)

The reset protection register, shown in Figure 4-12, is used to prevent unintended software reset requests caused by writes to the reset control register (RCR). The user should write the value 0x52535445 (RSTE in ASCII) to enable. Enable indication appears in the reset control enable register (RCER[CRE]). Reading this register always returns all zeros. To disable write to the reset control register (RCR), the user should write one to RCER[CRE].

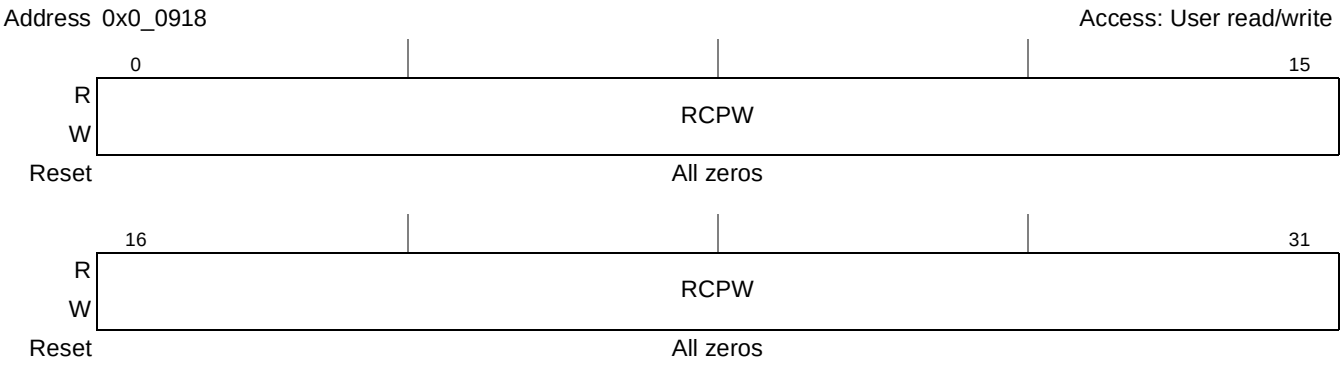


Figure 4-12. Reset Protection Register (RPR)

Table 4-30 defines the bit fields of RPR.

Table 4-30. RPR Bit Settings

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)

The 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.

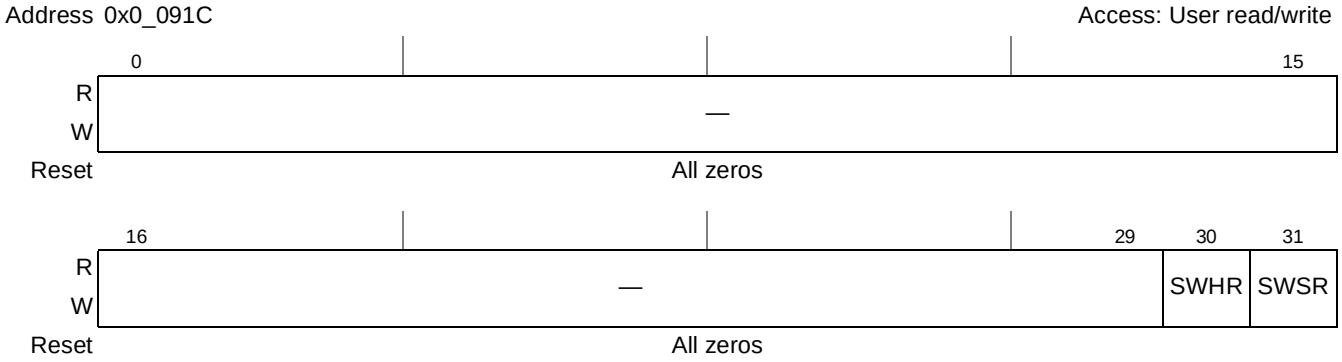


Figure 4-13. Reset Control Register (RCR)

Table 4-31 defines the bit fields of RCR.

Table 4-31. RCR Bit Settings

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

4.5.1.7 Reset Control Enable Register (RCER)

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

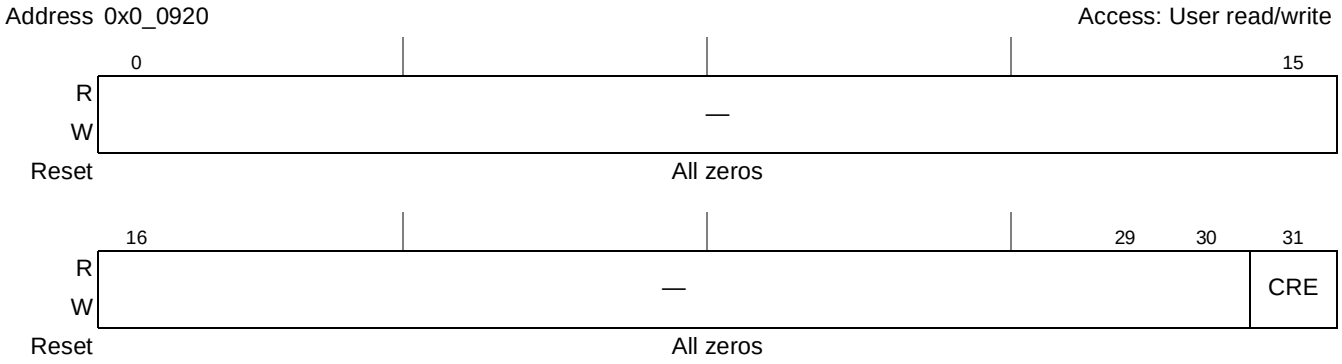


Figure 4-14. Reset Control Enable Register (RCER)

Table 4-32 defines the bit fields of RCER.

Table 4-32. 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 4-33.

Table 4-33. Clock Configuration Registers Memory Map

Address	Use	Access	Section/Page
0x0_0A00	System PLL mode register (SPMR)	R	4.5.2.1/4-39
0x0_0A04	Output clock control register (OCCR)	R/W	4.5.2.2/4-40
0x0_0A08	System clock control register (SCCR)	R/W	4.5.2.3/4-41
0x0_0A0C–0x0_0AFC	Reserved, should be cleared.	—	—

4.5.2.1 System PLL Mode Register (SPMR)

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

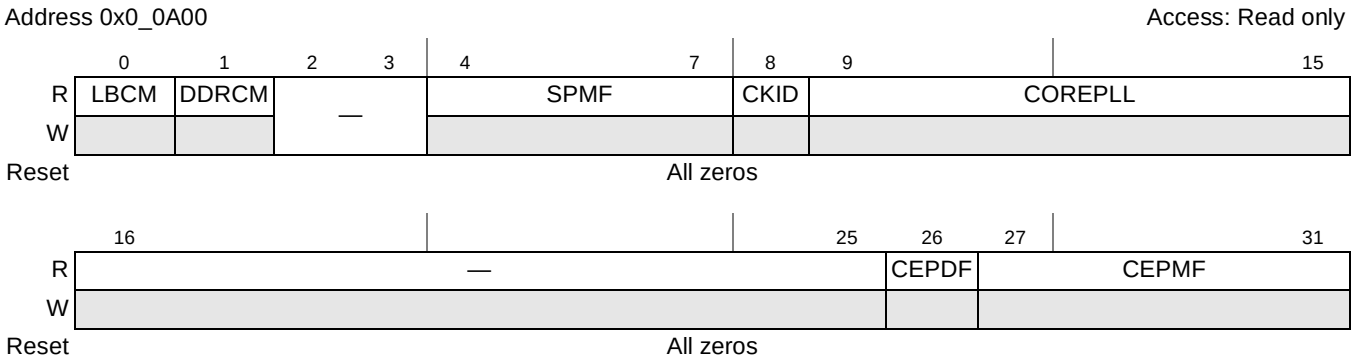


Figure 4-15. System PLL Mode Register

Table 4-34 defines the system PLL mode register bit fields.

Table 4-34. System PLL Mode Register Bit Settings

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

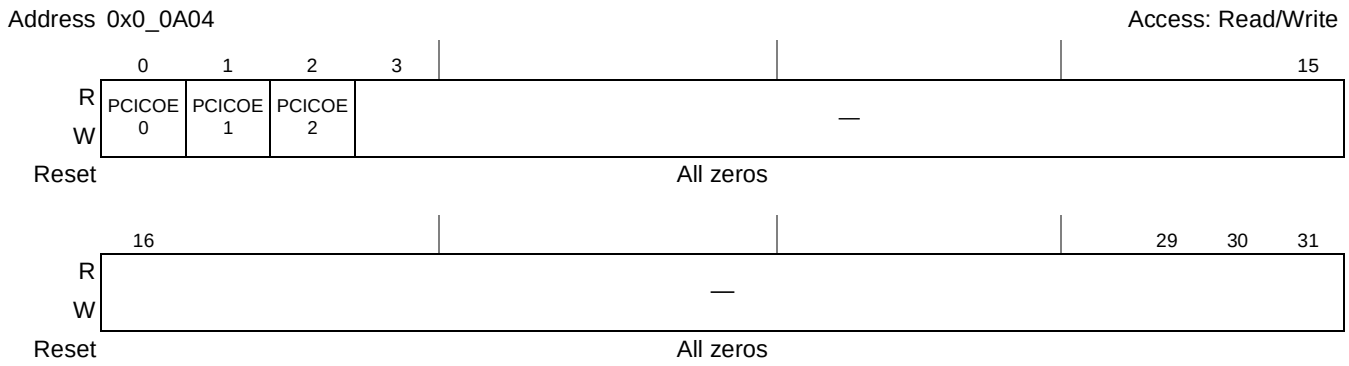


Figure 4-16. Output Clock Control Register (OCCR)

Table 4-35 defines the bit fields of OCCR.

Table 4-35. OCCR Bit Settings

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

4.5.2.3 System Clock Control Register (SCCR)

The system clock control register (SCCR), shown in Figure 4-17, controls device units that have a configurable clock ratio.

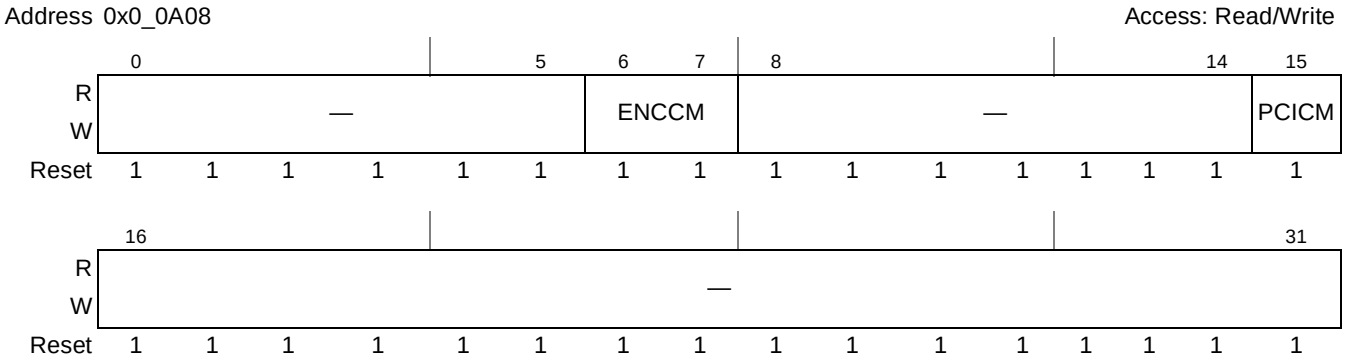


Figure 4-17. System Clock Control Register (SCCR)

Table 4-36 defines the bit fields of SCCR.

Table 4-36. SCCR Bit Settings

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6–7	ENCCM	Encryption core 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.

Table 4-36. SCCR Bit Settings (continued)

Bits	Name	Description
15	PCICM	PCI clock mode. Define the clock mode for all of the PCI complex - PCI and DMA. 0 PCI complex clocks are disabled. 1 PCI complex clocks are enabled.
16–31	—	Reserved

4.5.3 Clock Control DDR Registers

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 4-37. 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) **Note:** Offset 0x10 is a offset from the clock control DDR offset 0x001000	R/W	0xFC00_0000	4.5.3.1/4-42
0x14–0xFF	Reserved, should be cleared.	—	—	—

4.5.3.1 MCK Enable Register (MCKENR)

The MCK clock enable register (MCKENR), shown in Figure 4-18, enables or disables the DDR clock outputs.

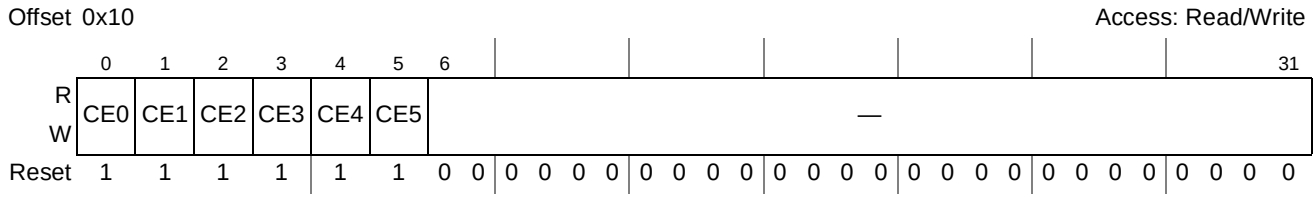


Figure 4-18. MCK Clock Register (MCKENR)

Table 4-38 describes MCKENR fields.

Table 4-38. MCKENR Field Description

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

Chapter 5
System Configuration

5.1 Introduction

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

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

5.2 Local Memory Map Overview and Example

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

The local memory map is defined by a set of eleven local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. 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	—

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

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

Figure 5-1 shows an example memory map.

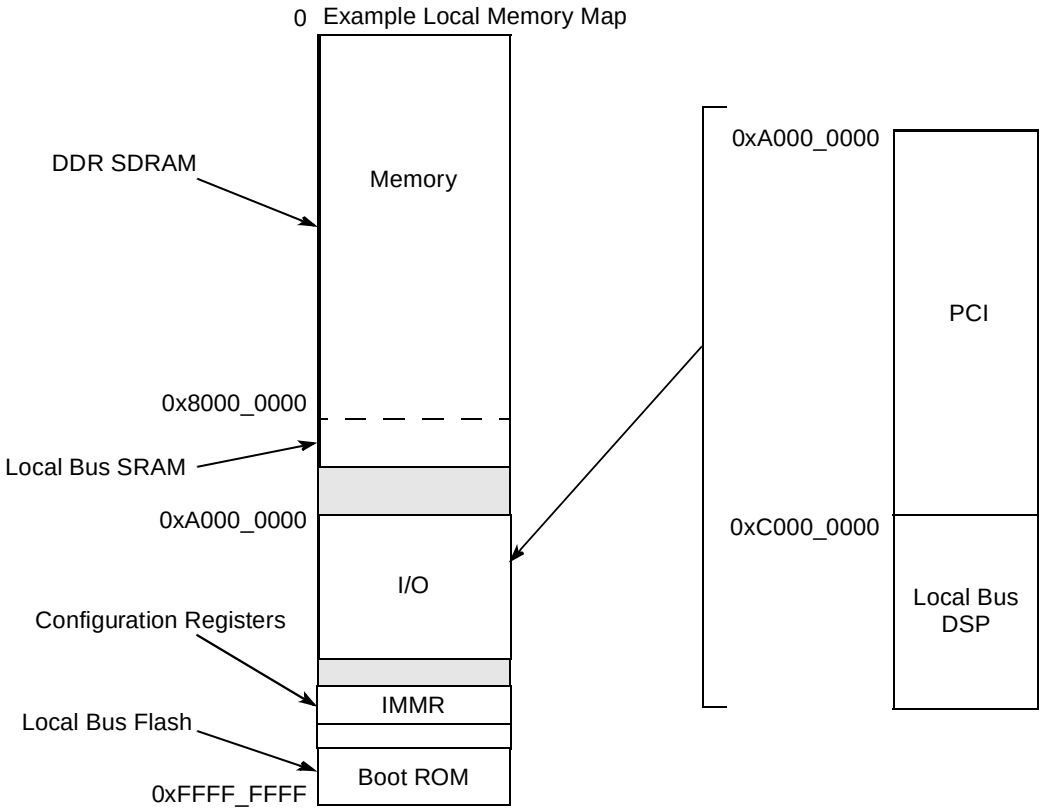


Figure 5-1. Local Memory Map Example

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

Table 5-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
7	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)

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

Window	Base Address	Size	Target Interface
1	0xFF80_0000	8 Mbytes	Local bus boot ROM Flash
4, 6, 8, 9, 10	Unused		

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

5.2.1 Address Translation and Mapping

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

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

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

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

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

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

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

Table 5-3. Format of Window Definitions

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

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

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

5.2.2 Window into Configuration Space

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

NOTE

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

5.2.3 Local Access Windows

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

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

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

5.2.3.1 Local Access Register Memory Map

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

Table 5-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	Internal memory map base address register (IMMRBAR)	R/W	0xFF40_0000	5.2.4.1/5-6
0x0_0004	Reserved	—	—	—
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	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	5.2.4.4/5-9
0x0_0040–0x0_005C	Reserved	—	—	—
0x0_0060	PCI local access window0 base address register (PCILAWBAR0)	R/W	0x0000_0000[3]	5.2.4.5/5-10
0x0_0064	PCI local access window0 attribute register (PCILAWAR0)	R/W	0x0000_0000[4]	5.2.4.6/5-11
0x0_0068	PCI local access window1 base address register (PCILAWBAR1)	R/W	0x0000_0000[5]	5.2.4.5/5-10
0x0_006C	PCI local access window1 attribute register (PCILAWAR1)	R/W	0x0000_0000	5.2.4.6/5-11
0x0_0070–0x0_009C	Reserved	—	—	—
0x0_00A0	DDR local access window0 base address register (DDRLAWBAR0)	R/W	0x0000_0000[6]	5.2.4.7/5-12
0x0_00A4	DDR local access window0 attribute register (DDRLAWAR0)	R/W	0x0000_0000[7]	5.2.4.8/5-13
0x0_00A8	DDR local access window1 base address register (DDRLAWBAR1)	R/W	0x0000_0000	5.2.4.7/5-12
0x0_00AC	DDR local access window1 attribute register (DDRLAWAR1)	R/W	0x0000_0000	5.2.4.8/5-13
0x0_00B0–0x0_00DC	Reserved	—	—	—
0x0_00E0	Secondary DDR local access window0 base address register (SDDRLAWBAR0)	R/W	0x0000_0000	5.2.4.9/5-14
0x0_00E4	Secondary DDR local access window0 attribute register (SDDRLAWAR0)	R/W	0x0000_0000	5.2.4.10/5-14
0x0_00E8	Secondary DDR local access window1 base address register (SDDRLAWBAR1)	R/W	0x0000_0000	5.2.4.9/5-14
0x0_00EC	Secondary DDR local access window1 attribute register (SDDRLAWAR1)	R/W	0x0000_0000	5.2.4.10/5-14

[1] Depends on reset configuration word high values. See Section 5.2.4.3.1, “LBLAWBAR0[BASE_ADDR] Reset Value,” for details.

[2] Depends on reset configuration word high values. See Section 5.2.4.4.1, “LBLAWAR0[EN] and LBLAWAR0[SIZE] Reset Value,” for details.

[3] Depends on reset configuration word high values. See Section 5.2.4.5.1, “PCILAWBAR0[BASE_ADDR] Reset Value” for details.

[4] Depends on reset configuration word high values. See Section 5.2.4.7.1, “DDRLAWBAR0[BASE_ADDR] Reset Value,” for details.

[5] Depends on reset configuration word high values. See Section 5.2.4.6.1, “PCILAWAR0[EN] and PCILAWAR0[SIZE] Reset Value,” for details.

[6] Depends on reset configuration word high values. See Section 5.2.4.7.1, “DDRLAWBAR0[BASE_ADDR] Reset Value,” for details.

[7] Depends on reset configuration word high values. See Section 5.2.4.8.1, “DDRLAWAR0[EN] and DDRLAWAR0[SIZE] Reset Value,” for details.

5.2.4 Local Access Register Descriptions

5.2.4.1 Internal Memory Map Registers Base Address Register (IMMRBAR)

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

5.2.4.1.1 Updating IMMRBAR

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

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

The IMMRBAR is shown in Figure 5-2.

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

Table 5-5 defines the bit fields of IMMRBAR.

Table 5-5. IMMRBAR Bit Settings

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

5.2.4.2 Alternate Configuration Base Address Register (ALTCBAR)

The alternate configuration base address register (ALTCBAR) is used to define the base address for an alternate 1-Mbyte region of configuration space to be used by the boot sequencer. By loading the proper boot sequencer command in the serial ROM, the base address in the ALTCBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 32-bit address. Thus, by configuring this register, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 17.4.5, "Boot Sequencer Mode," of Chapter 17, "I^2C Interfaces," 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.

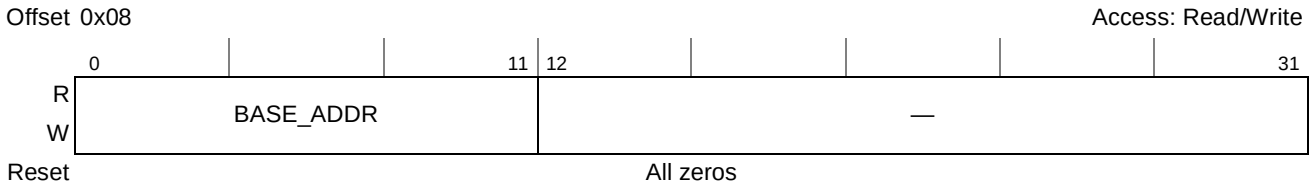


Figure 5-3. Alternate Configuration Base Address Register (ALTCBAR)

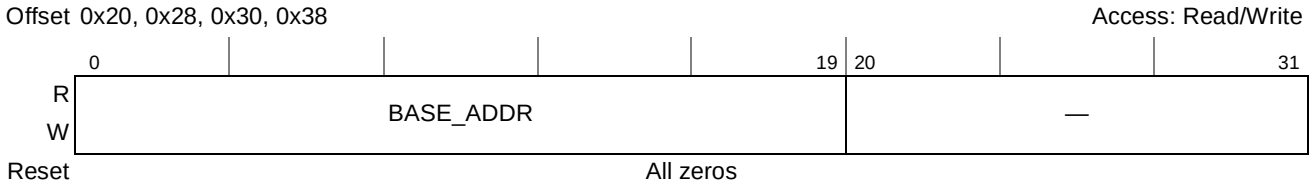
Table 5-6 defines the bit fields of ALTCBAR.

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.

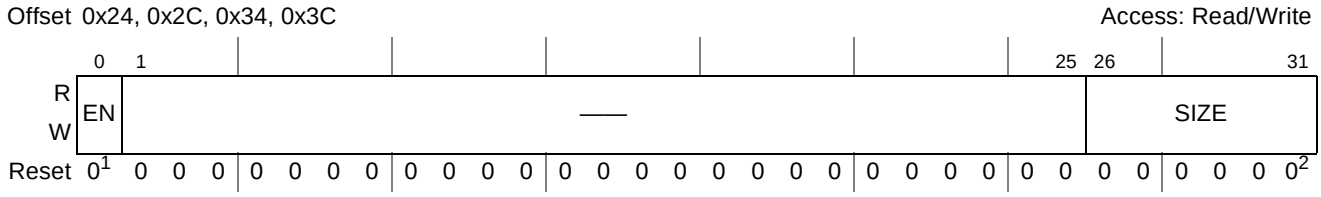
Table 5-8 defines the reset value of LBLAWBAR0[BASE_ADDR].

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.



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. 000000–001010Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 00110116 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

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

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

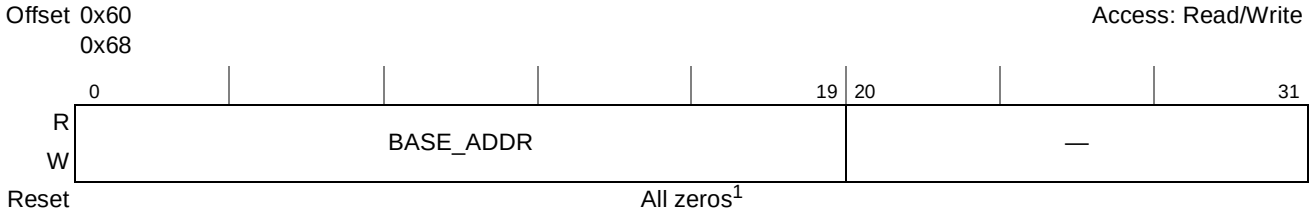
Table 5-10 defines the reset value for LBLAWAR0[EN].

Table 5-10. LBLAWAR0[EN] Reset Value

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

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

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



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

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

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

Table 5-11. PCILAWBAR0–PCILAWBAR1 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window n. The specified base address should be aligned to the window size, as defined by PCILAWARn[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.

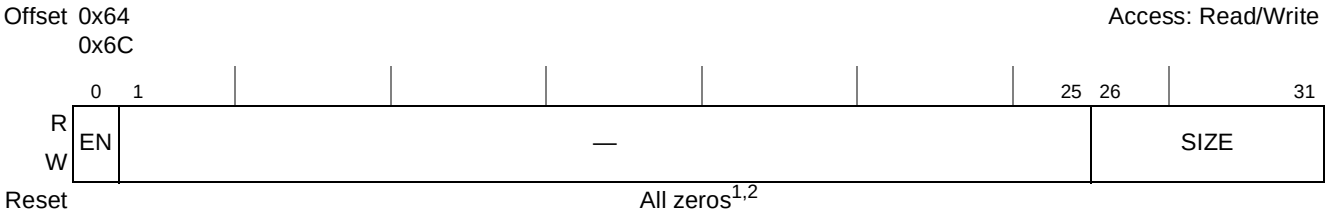
Table 5-12 defines the reset value of PCILAWBAR0[BASE_ADDR].

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. 000000–001010Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 00110116 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

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

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

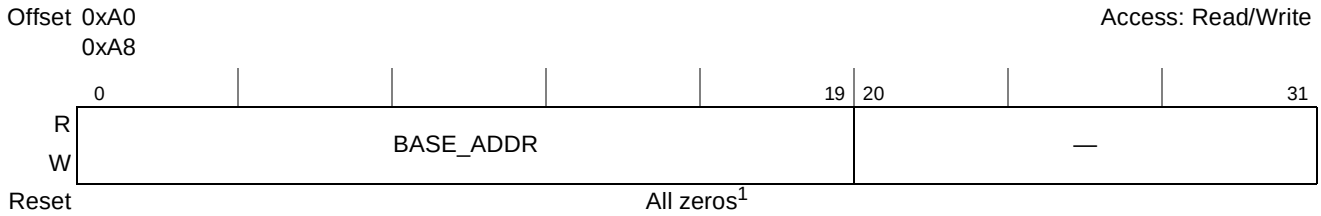
Table 5-14 defines the reset value of PCILAWAR0[EN].

Table 5-14. PCILAWAR0[EN] Reset Value

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

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

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



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

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

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

Table 5-15. DDRLAWBAR0–DDRLAWBAR1 Bit Settings

Bits	Name	Description
0–19	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window *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.

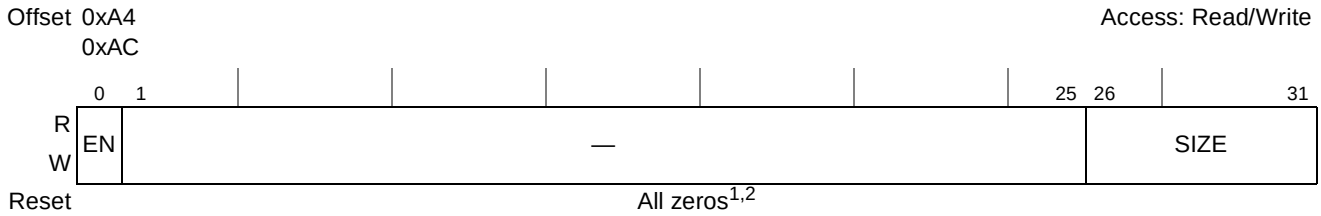
Table 5-16 defines the reset value DDRLAWBAR0.

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. 000000–001010Reserved. Window is undefined. 001011 4 Kbytes 001100 8 Kbytes 00110116 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved. Window is undefined.

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

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

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

Table 5-18. DDRLAWAR0[EN] Reset Value

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

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

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

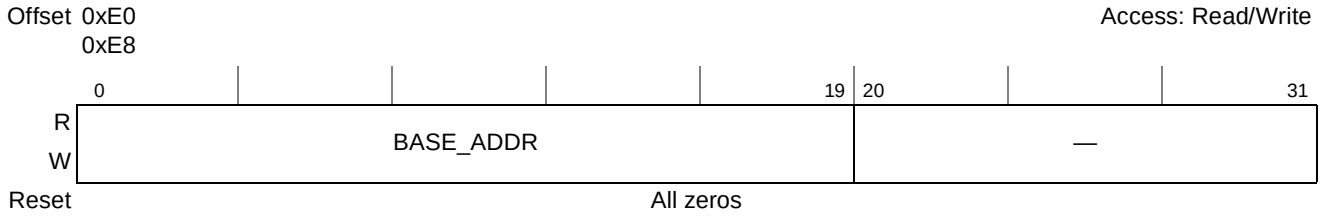


Figure 5-10. Secondary DDR Local Access Window n Base Address Registers (SDDRLAWBAR0–SDDRLAWBAR1)

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

Table 5-19. SDDRLAWBAR0–SDDRLAWBAR1 Bit Settings

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

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

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

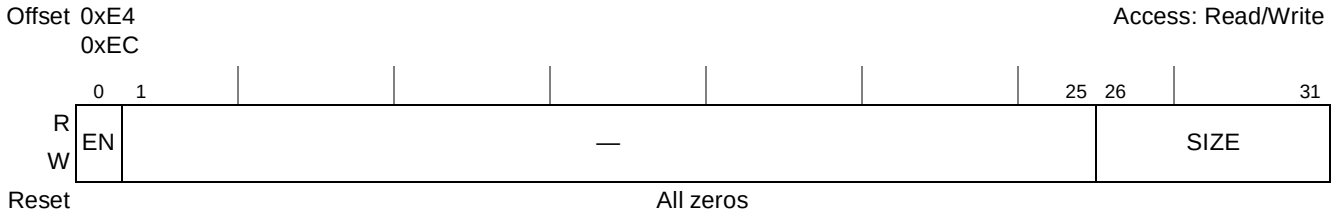


Figure 5-11. Secondary DDR Local Access Window n Attributes Registers (SDDRLAWAR0–SDDRLAWAR1)

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

Table 5-20. SDDRLAWAR0–SDDRLAWAR1 Bit Settings

Bits	Name	Description
0	EN	0 The secondary DDR local access window n is disabled. 1 The secondary DDR local access window n is enabled and other SDDRLAWARn and SDDRLAWBARn fields combine to identify an address range for this window.
1–25	—	Reserved. Write has no effect, read returns 0.
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010Reserved. Window is undefined. 0010114 Kbytes 0011008 Kbytes 00110116 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 0111102 Gbytes 011111–111111Reserved. Window is undefined.

5.2.5 Precedence of Local Access Windows

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

Table 5-21. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x7FF0_0000	1 Mbyte	Local bus
7	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 e300c1 core, the read of LBLAWAR3 should be followed by an **isync** instruction.

5.2.7 Distinguishing Local Access Windows from Other Mapping Functions

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

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

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

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

5.3.1 QUICC Engine Secondary Bus Access Windows Register Memory Map

Table 5-22 shows the memory map for the QUICC Engine secondary bus access windows registers.

NOTE

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

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

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00000	Local bus memory controller start address register (LBMCSAR)	R/W	0x0000_0000	5.3.2.1/55-18
0x00004	Secondary DDR memory controller start address register (SDMCSAR)	R/W	0x0000_0000	5.3.2.2/55-19
0x00008–0x0003C	Reserved			
0x00040	Local bus memory controller end address register (LBMCEAR)	R/W	0x0000_0000	5.3.2.3/55-19
0x00044	Secondary DDR memory controller end address register (SDMCEAR)	R/W	0x0000_0000	5.3.2.4/55-20
0x00048–0x0007C	Reserved			
0x00080	Local bus memory controller attributes register (LBMCAR)	R/W	0x0000_0000	5.3.2.5/55-20
0x00084	Secondary DDR memory controller attributes register (SDMCAR)	R/W	0x0000_0000	5.3.2.6/55-21
0x00088–0x007FC	Reserved			

5.3.2 QUICC Engine Secondary Bus Access Windows Registers

5.3.2.1 Local Bus Memory Controller Start Address Register (LBMCSAR)

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

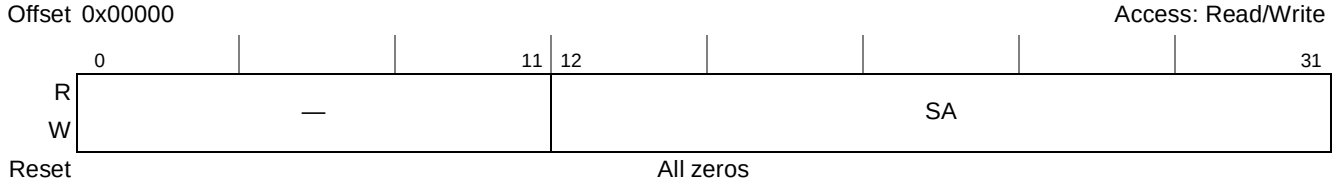


Figure 5-12. Local Bus Memory Controller Start Address Register (LBMCSAR)

Table 5-34 defines the bit fields of LBMCSAR.

Table 5-23. LBMCSAR Bit Settings

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

5.3.2.2 Secondary DDR Memory Controller Start Address Register (SDMCSAR)

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

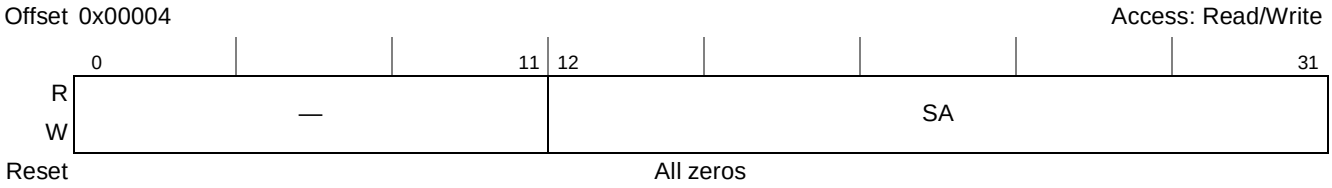


Figure 5-13. Secondary DDR Memory Controller Start Address Register (SDMCSAR)

Table 5-34 defines the bit fields of SDMCSAR.

Table 5-24. SDMCSAR Bit Settings

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

5.3.2.3 Local Bus Memory Controller End Address Register (LBMCEAR)

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

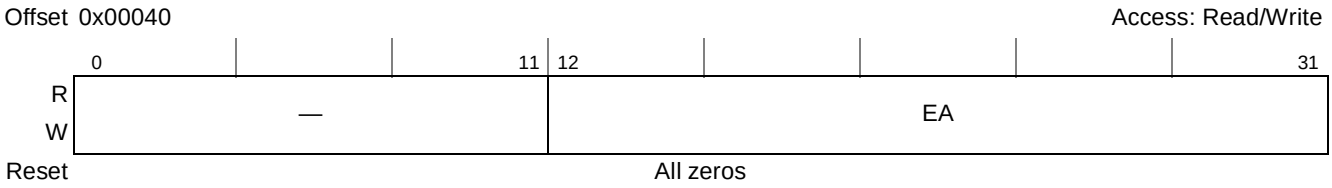


Figure 5-14. Local Bus Memory Controller End Address Register (LBMCEAR)

Table 5-34 defines the bit fields of LBMCEAR.

Table 5-25. LBMCEAR Bit Settings

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

5.3.2.4 Secondary DDR Memory Controller End Address Register (SDMCEAR)

The secondary DDR memory controller end address register, shown in Figure 5-15, defines the end address of the memory region which is assigned to the secondary DDR memory controller in the QUICC Engine block's local bus. To ensure proper operation do not modify this register while the specific window is enabled. (First clear SDMCAR[WEN].)

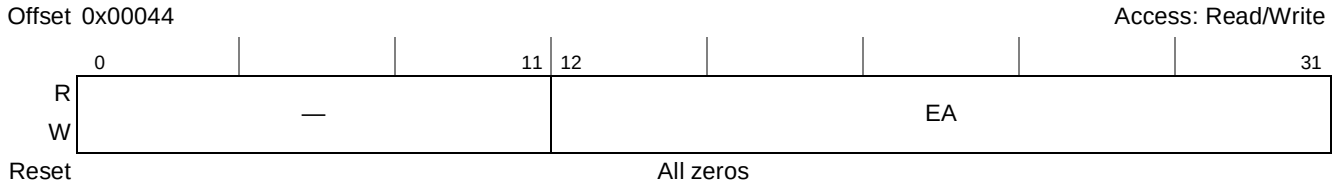


Figure 5-15. Secondary DDR Memory Controller End Address Register (SDMCEAR)

Table 5-34 defines the bit fields of SDMCEAR.

Table 5-26. SDMCEAR Bit Settings

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

5.3.2.5 Local Bus Memory Controller Attributes Register (LBMCAR)

The local bus memory controller attributes register, shown in Figure 5-16, defines the attributes of the memory region which is assigned to the local bus memory controller in the QUICC Engine block's local bus.

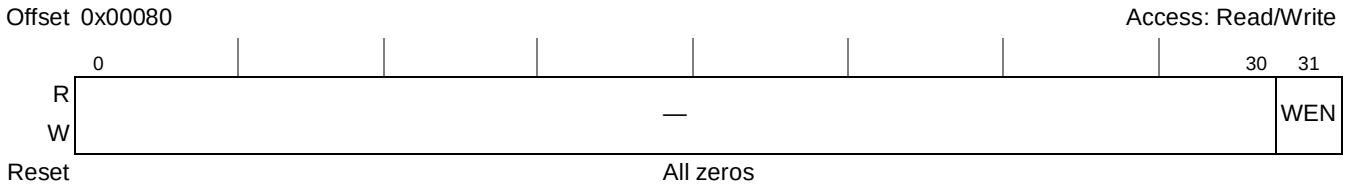


Figure 5-16. Local Bus Memory Controller Attributes Register (LBMCAR)

Table 5-34 defines the bit fields of LBMCAR.

Table 5-27. LBMCAR Bit Settings

Bits	Name	Description
0–30	Reserved	Reserved should be cleared.
31	WEN	0 Transaction on the QUICC Engine block's local bus will not be forwarded to the local bus memory controller. 1 Transaction on the QUICC Engine block's local bus which hits in the memory region defined by LBMCSAR and LBMCEAR is forwarded to the local bus memory controller.

5.3.2.6 Secondary DDR Memory Controller Attributes Register (SDMCAR)

The secondary DDR memory controller attributes register shown in Figure 5-17 defines the attributes of the memory region which is assigned to the secondary DDR memory controller in QUICC Engine block's local bus.

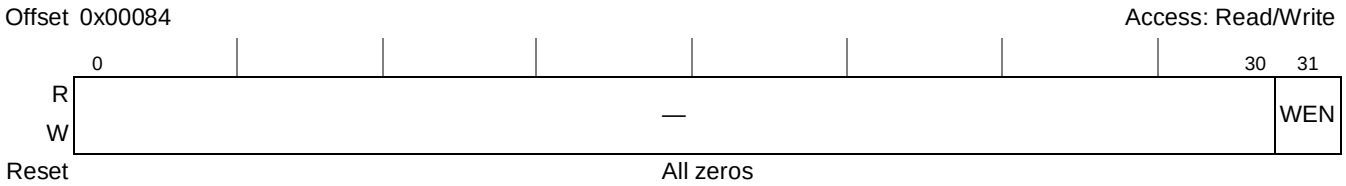


Figure 5-17. Secondary DDR Memory Controller Attributes Register (SDMCAR)

Table 5-34 defines the bit fields of SDMCAR.

Table 5-28. SDMCAR Bit Settings

Bits	Name	Description
0–30	Reserved	Reserved should be cleared.
31	WEN	0 Transaction on QUICC Engine block's local bus will not be forwarded to the secondary DDR memory controller. 1 Transaction on QUICC Engine block's local bus which hits in the memory region defined by SDMCSAR and SDMCEAR is forwarded to the secondary DDR memory controller.

5.3.3 QUICC Engine Secondary Bus Windows Operation Description

The QUICC Engine secondary bus windows manage the forwarding of transactions from the QUICC Engine block's local bus to a specific interface connected to this bus. Two specific interfaces are connected to the QUICC Engine block's local bus: the local bus memory controller and the secondary DDR SDRAM memory controller. Note that these two interfaces are accessible from the coherent system bus too through the local access windows (see Section 5.2, "Local Memory Map Overview and Example").

Two QUICC Engine secondary bus windows are available. Each of these windows is associated to one of the interfaces and defines a certain memory region. The address of each transaction on the QUICC Engine block's local bus is compared to the memory regions defined by the QUICC Engine secondary bus windows and forwarded to the region which contains this address. In case there is no matching region a transfer error is reported to the QUICC Engine block. In case there is an overlap between the two memory regions, transactions hitting in the overlapping region is forwarded to the secondary DDR memory controller. Thus the secondary DDR memory controller has higher priority. This feature can be used to generate a secondary DDR memory controller region in the middle of a large local bus memory controller region as described in Section 5.3.4, "QUICC Engine Secondary Bus Windows Example."

The local bus memory controller and the secondary DDR memory controller are both accessible from the coherent system bus (through local access windows) and from the QUICC Engine block's local bus (through QUICC Engine secondary bus windows). It is theoretically possible to allow access to one of these interfaces only from a certain bus. It is the users responsibility to correctly configure the two window mechanisms according to his system's needs.

5.3.4 QUICC Engine Secondary Bus Windows Example

Figure 5-18 shows an example memory map and the usage of QUICC Engine secondary bus windows.

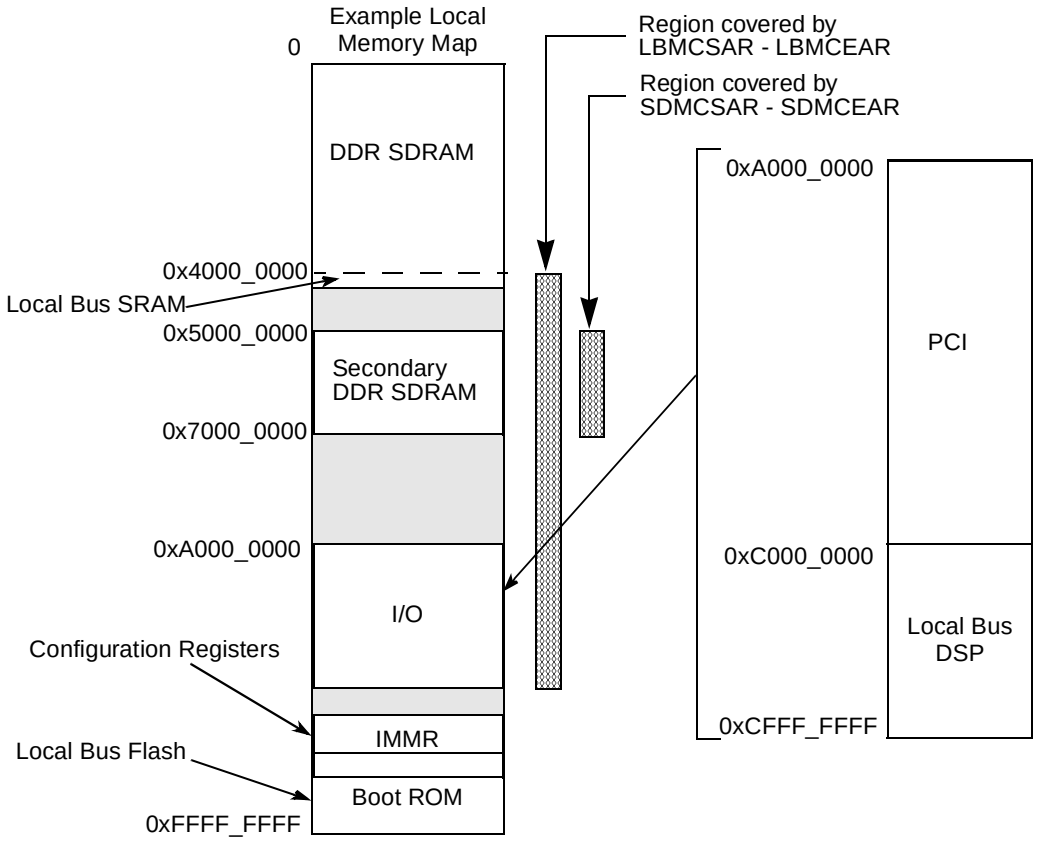


Figure 5-18. Local Memory Map Example

Table 5-30 shows one corresponding set of local access window settings.

Table 5-29. Local Access Windows Example

Window	Base Address	Size	Target Interface
7	0x0000_0000	1 Gbyte	DDR SDRAM
2	0x4000_0000	1 Mbyte	Local bus (SRAM)
9	0x5000_0000	256 Mbyte	Secondary DDR SDRAM
5	0xA000_0000	512 Mbytes	PCI
3	0xC000_0000	256 Mbytes	Local bus (DSP)
0	0xFF40_0000	2 Mbytes	Configuration registers (IMMR)
1	0xFF80_0000	8 Mbytes	Local bus boot ROM flash
4,6,8,10	Unused		

Table 5-30 shows the settings of QUICC Engine secondary bus windows.

Table 5-30. QUICC Engine Secondary Bus Windows Settings

Window	Value in Start Address Register	Value in End Address Register
Local bus memory controller	0x4000_0000	0xCFFF_F000
Secondary DDR memory controller	0x5000_0000	0x5FFF_F000

In this example the system includes several interfaces/devices: DDR SDRAM on the coherent system bus, DDR SDRAM on local bus, SRAM, PCI, DSP device, Boot ROM flash and the configuration registers (IMMR). All these devices are accessible through the coherent system bus. A subset of all these devices is accessible by the QUICC Engine block's local bus: DDR SDRAM on local bus, SRAM and the DSP device. Note that the PCI is not accessible by the QUICC Engine secondary bus although it is within the region of local bus memory controller window. This is because the PCI is connected to the coherent system bus and not as one of the banks of the local bus memory controller. Thus, to access the PCI interface the QUICC Engine block should initiate the transaction on its coherent system bus. Note that transactions hitting in the region 0x5000_0000– 0x5FFF_FFFF is directed to the secondary DDR memory controller although the window of the local bus memory controller is overlapping on this region. This is due to the precedence of secondary DDR memory controller window which is used intentionally in this example.

5.4 System Configuration

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

5.4.1 External Signal Description

Section 5.4.1.1, "PCI_MODE Signal," describes the MPC8360E external signal that affects the system configuration.

5.4.1.1 $\overline{\text{PCI\_MODE}}$ Signal

One signal affects the system configuration of the MPC8360E. Table 5-31 lists this signal.

Table 5-31. System Configuration Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
$\overline{\text{PCI\_MODE}}$	$\overline{\text{PCI\_MODE}}$	Global enable for PCI external I/O signals	I	N/A	—

Table 5-32 provides a detailed description of the external MPC8360E system configuration signal.

Table 5-32. System Configuration Signal—Detailed Description

Signal	I/O	Description	
PCI_MODE	I	A group of QUICC Engine parallel I/O ports (CE_PF[3:29] & CE_PG[0:31]) may function as PCI signals. The PCI_MODE signal enables PCI mode for these signals. If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.	
		State Meaning	• Asserted—QUICC Engine parallel I/O ports CE_PF[3:29] & CE_PG[0:31] function as PCI signals. For CE_PF[26:28] which may function as PCI_CLK_OUT[0:2], the PCI function is selected with the reset configuration bit RCWH[PCICKDRV] (in addition to assertion of the PCI_MODE signal). See Section 4.3.2.2, "Reset Configuration Word High Register (RCWHR)." Assertion of PCI_MODE during PORESET flow affects the reset value of QUICC Engine parallel I/O ports configuration registers and selects the PCI function. In addition, the PCI function is selected during normal operation regardless of the values in these configuration registers. Thus the user may change the values in the configuration registers, but the PCI function will still be selected as long as PCI_MODE is asserted. • Negated—QUICC Engine parallel I/O ports CE_PF[3:29] & CE_PG[0:31] function according to the values in their configuration registers (this may be PCI or non-PCI functionality.)
		Timing	This input signal is sampled during the assertion of PORESET, after a stable clock is supplied (PORESET flow). During normal operation, the state of this signal affects the QUICC Engine ports functionality asynchronously.

5.4.2 System Configuration Register Memory Map

Table 5-33 shows the memory map for the system configuration registers.

Table 5-33. System Configuration Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00100	System general purpose register low (SGPRL)	R/W	0x0000_0000	5.4.3.1/5-25
0x00104	System general purpose register high (SGPRH)	R/W	0x0000_0000	5.4.3.2/5-25
0x00108	System part and revision ID register (SPRIDR)	R		5.4.3.3/5-26
0x0010C	Reserved	—	—	—
0x00110	System priority configuration register (SPCR)	R/W	0x0000_0000	5.4.3.4/5-27
0x00114	System I/O configuration register low (SICRL)	R/W	0x0000_0000[1]	5.4.3.5/5-28
0x00118	System I/O configuration register high (SICRH)	R/W	0x0000_0000	5.4.3.6/5-30
0x0011C–0x001FC	Reserved	—	—	—
0x00128	DDR control driver register (DDRCDR)	R/W		5.4.3.8/5-33
0x0012C	DDR debug status register (DDRDSR)	R	0x3300_0000	5.4.3.9/5-35

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

5.4.3 System Configuration Registers

5.4.3.1 System General Purpose Register Low (SGPRL)

The system general purpose register low (SGPRL), shown in Figure 5-19, can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

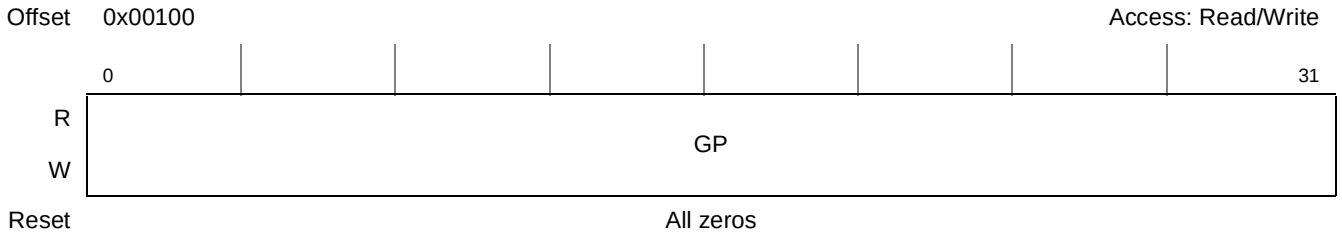


Figure 5-19. System General Purpose Register Low (SGPRL)

Table 5-34 defines the bit fields of SGPRL.

Table 5-34. SGPRL Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.4.3.2 System General Purpose Register High (SGPRH)

The system general purpose register high (SGPRH), shown in Figure 5-20, can be used by software for any purpose. The values set in this register have no effect on the internal hardware.

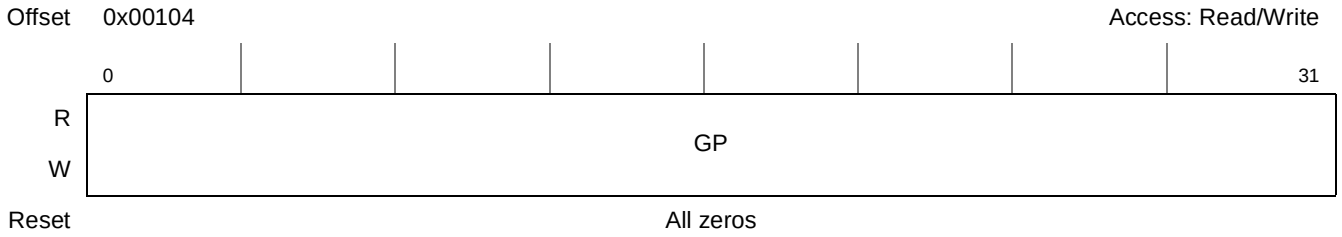


Figure 5-20. System General Purpose Register High (SGPRH)

Table 5-35 defines the bit fields of SGPRH.

Table 5-35. SGPRH Bit Settings

Bits	Name	Description
0–31	GP	General purpose

5.4.3.3 System Part and Revision ID Register (SPRIDR)

SPRIDR, shown in Figure 5-21, provides information about the device and revision numbers.

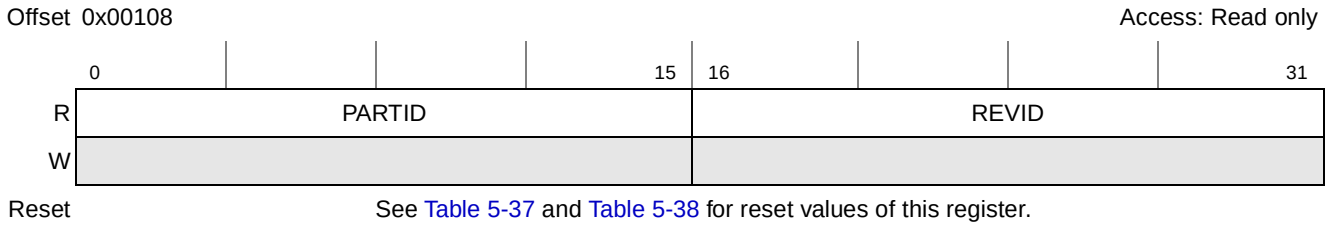


Figure 5-21. System Part and Revision ID Register (SPRIDR)

Table 5-36 defines the bit fields of SPRIDR.

Table 5-36. SPRIDR Bit Settings

Bits	Name	Description
0–15	PARTID	Part identification. This read-only field is mask-programmed with a code corresponding to the device number. It is intended to help factory test and user code that is sensitive to device changes. The device number changes according to manufacturing considerations. See Table 5-37 for values of this field.
16–31	REVID	Revision identification. This read-only field is mask-programmed with a code corresponding to the revision number of the part defined in PARTID field. It is intended to help factory test and user code that is sensitive to device changes. The mask number is programmed in a commonly changed layer, and changes with each mask set change. See Table 5-38 for values of this field.

5.4.3.3.1 SPRIDR[PARTID] Coding

Table 5-37 defines the reset values of SPRIDR[PARTID].

Table 5-37. PARTID Coding

PARTID	Device Name	Package Type
0x8048	MPC8360E	TBGA
0x8049	MPC8360	TBGA
0x804A	MPC8358E	TBGA
0x804B	MPC8358	TBGA
0x804E	MPC8358E	PBGA
0x804F	MPC8358	PBGA

Table 5-38 defines the reset values of SPRIDR[REVID].

Table 5-38. REVID Coding

REVID	Device Revision
0x0010	1.0
0x0011	1.1

Table 5-38. REVID Coding (continued)

REVID	Device Revision
0x0012	1.2
0x0020	2.0

5.4.3.4 System Priority and Configuration Register (SPCR)

The system priority and configuration register (SPCR), shown in Figure 5-22, controls the priority of requests for transactions on the internal system bus. This priority is considered by the system arbiter whenever an internal unit requests mastership of the coherent system bus (CSB). The SPCR also includes some other control functions.

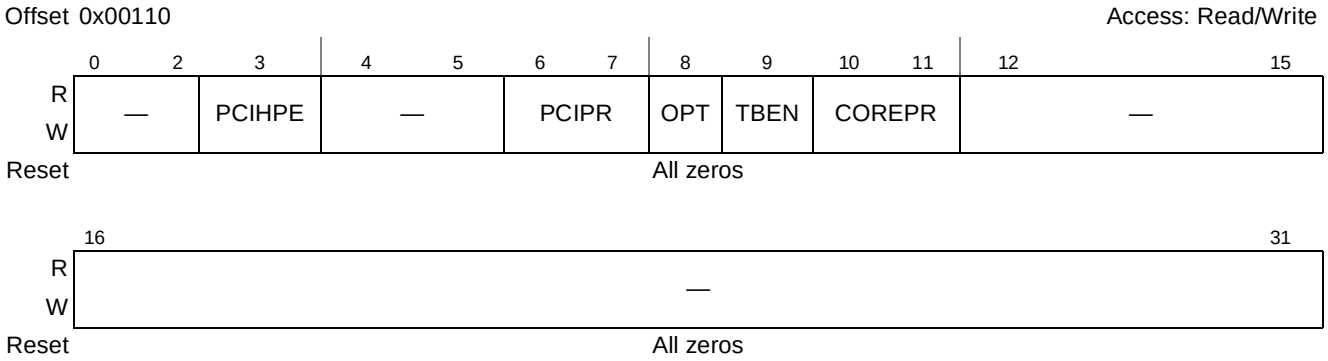


Figure 5-22. System Priority Configuration Register (SPCR)

Table 5-39 defines the bit fields of SPCR.

Table 5-39. SPCR Bit Settings

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

Table 5-39. SPCR Bit Settings (continued)

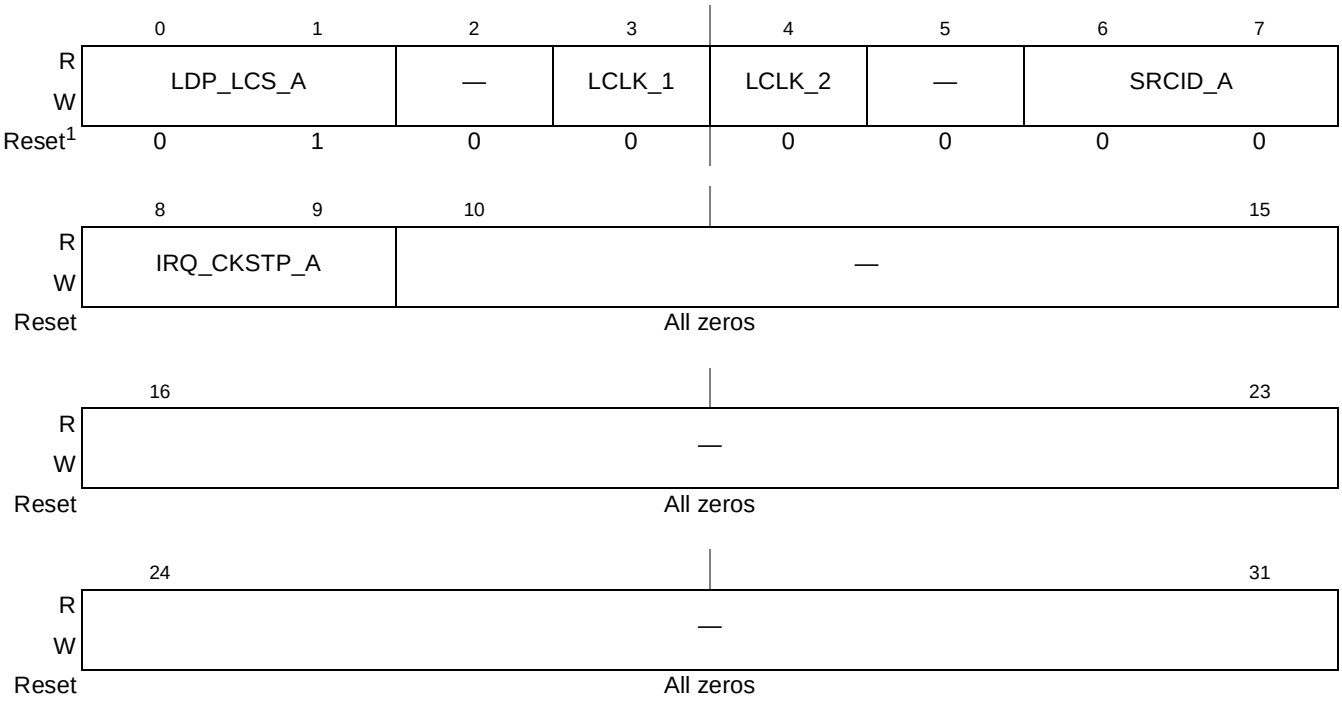
Bits	Name	Description
8	OPT	Optimize. Setting this bit may enhance the performance of transactions issued to the internal coherent system bus (CSB) by the security engine (SEC) and the QUICC Engine block. Performance is enhanced by reading more bytes on the bus than actually needed by the master in the case that this is more efficient. The user may set this bit only if it is known that QUICC /Engine SEC transactions sent to the internal CSB are not accessing devices in which speculative reads may change the state of the device (for example, FIFOs in which reading a byte may advance some internal counter). 0 No performance enhancement. 1 Performance enhancement by speculative reading is enabled.
9	TBEN	e300c1 core time base unit enable 0 Time base unit is disabled. 1 Time base unit is enabled.
10–11	COREPR	e300c1 core CSB request priority. The priority level for the core in accessing the CSB can be chosen from 4 possible levels. 00 Level 0 (lowest priority) 01 Level 1 10 Level 2 11 Level 3 (highest priority)
12–31	—	Reserved. Should be cleared.

5.4.3.5 System I/O Configuration Register Low (SICRL)

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

Figure 5-23 shows SICRL.

Offset 0x00114 Access: Read/Write



[1] The reset value of SICRL[0] depends on the value of RCWH[30] which is loaded into the reset configuration word.

Figure 5-23. System I/O Configuration Register Low (SICRL)

Table 5-40 defines the bit fields of SICRL. Each Pin Function column lists the name of the multi-function pin used in this option. Some groups have only two options (shown as Pin Function 0 and Pin Function 1) and therefore only one control bit. In this case they can only have a value of 0b0 or 0b1. Other groups may have four options (shown as Pin Function 0, Pin Function 1, Pin Function 2, and Pin Function 3), and therefore two control bits. In this case they can have a value of 0b00, 0b01, 0b10 or 0b11. Use the notations '0b*N*' or '0b*NN*' according to whether a group has one or two control bits, respectively.

Table 5-40. SICRL Bit Settings

SICRL[Bits] Value		0b0/0b00	0b1/0b01	0b10	0b11
Bits	**Group**	**Pin Function 0**	**Pin Function 1**	**Pin Function 2**	**Pin Function 3**
0–1	LDP_LCS_A	$\overline{\text{CKSTOP\_OUT}}$	LDP[0]	N/A	N/A
		$\overline{\text{CKSTOP\_IN}}$	LDP[1]	N/A	N/A
		$\overline{\text{LCS}}$[6]	LDP[2]	$\overline{\text{LCS}}$[6]	N/A
		$\overline{\text{LCS}}$[7]	LDP[3]	$\overline{\text{LCS}}$[7	N/A
2	Reserved	—	—	—	—
3	LCLK_1	LCLK1	$\overline{\text{LCS}}$[6]	N/A	N/A
4	LCLK_2	LCLK2	$\overline{\text{LCS}}$[7]	N/A	N/A

Table 5-40. SICRL Bit Settings (continued)

SICRL[Bits] Value		0b0/0b00	0b1/0b01	0b10	0b11
Bits	**Group**	**Pin Function 0**	**Pin Function 1**	**Pin Function 2**	**Pin Function 3**
5	Reserved	—	—	—	—
6–7	SRCID_A	UART1_SOUT	M1SRCID0	M2SRCID0	LSRCID0
		UART1_SIN	M1SRCID1	M2SRCID1	LSRCID1
		$\overline{\text{UART1\_CTS}}$	M1SRCID2	M2SRCID2	LSRCID2
		$\overline{\text{UART1\_RTS}}$	M1SRCID3	M2SRCID3	LSRCID3
		$\overline{\text{IRQ1}}$	M1SRCID4	M2SRCID4	LSRCID4
		$\overline{\text{IRQ2}}$	M1DVAL	M2DVAL	LDVAL
8–9	IRQ_CKSTP_A	$\overline{\text{IRQ6}}$	N/A	$\overline{\text{LCS}}$[6]	$\overline{\text{CKSTOP\_OUT}}$
		$\overline{\text{IRQ7}}$	N/A	$\overline{\text{LCS}}$[7]	$\overline{\text{CKSTOP\_IN}}$
10–31	Reserved	—	—	—	—

5.4.3.6 System I/O Configuration Register High (SICRH)

The system I/O configuration register high, shown in Figure 5-24, controls the multiplexing of the rest of the device I/O pins. Each bit or set of bits in this register select which function is used by a certain group of the device pins. The reset value of bit 2 in this register (SDDRIOE group) depends on the SDDRIOE field setting in the reset configuration word high.

Offset 0x00118 Access: Read/Write

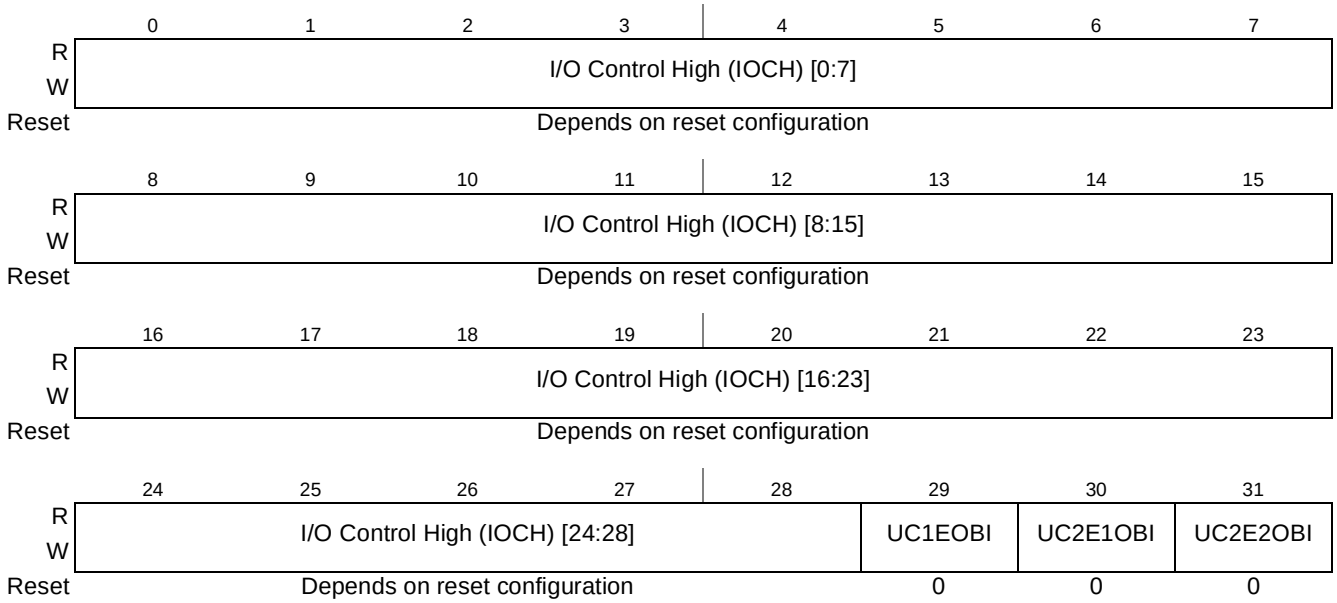


Figure 5-24. System I/O Configuration Register High (SICRH)

Table 5-41 defines the bit fields of SICRH. Each Pin Function column lists the name of the multi-function pin used in this option. Some groups have only two options (shown as Pin Function 0 and Pin Function 1) and therefore only one control bit. In this case they can have the value of 0b0 or 0b1 only. Other groups may have three options (shown as Pin Function 0, Pin Function 1, and Pin Function 2), and therefore two control bits. In this case they can have the value of 0b00, 0b01 or 0b10. A value of 0b11 is illegal for all groups. Use the 0b*N* or 0b*NN* notations according to a group having one or two control bits respectively.

Table 5-41 defines the bit fields of SICRH. Each Pin Function column lists the name of the multi-function pin used in this option. Each group has two options (shown as Pin Function 0 and Pin Function 1) and therefore one control bit, that can have the value of 0 or 1 only.

Note that bits 29, 30 and 31 (UC1EOBI, UC2E1OBI and UC2E2OBI), which control UCC Ethernet output buffer impedance, are described in Table 5-42.

Table 5-41. SICRH Bit Settings

SICRH[Bits] Value:		0b0	0b1	Reset Value
Bits	**Group**	**Pin Function 0**	**Pin Function 1**	
0	DDR	MEMC1_MECC[0]	MEMC1_MSRCID[0]	0
		MEMC1_MECC[1]	MEMC1_MSRCID[1]	
		MEMC1_MECC[2]	MEMC1_MSRCID[2]	
		MEMC1_MECC[3]	MEMC1_MSRCID[3]	
		MEMC1_MECC[4]	MEMC1_MSRCID[4]	
		MEMC1_MECC[5]	MEMC1_MDVAL	
1	Secondary DDR	MEMC2_MECC[0]	MEMC2_MSRCID[0]	0
		MEMC2_MECC[1]	MEMC2_MSRCID[1]	
		MEMC2_MECC[2]	MEMC2_MSRCID[2]	
		MEMC2_MECC[3]	MEMC2_MSRCID[3]	
		MEMC2_MECC[4]	MEMC2_MSRCID[4]	
		MEMC2_MECC[5]	MEMC2_MDVAL	
2	SDDRIOE	MEMC1_MDQ[32:63]	MEMC2_MDQ[0:31]	SDDRIOE bit from reset configuration word high.
3	IRQ3	$\overline{\text{IRQ}}$[3]	$\overline{\text{CORE\_SRESET}}$	0
4–29	Reserved	—	—	—

Table 5-42. SICRH[29–31] Bit Settings

Bits	Name	Description	Reset Value
29	UC1EOBI	UCC1 Ethernet output buffer impedance. This bit controls the output buffer impedance of UCC1 output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC1 I/O pins (LVDD0). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts. See Section 3.4.7, "RGMII Pins," for details on reduced pin mode interface.	0
30	UC2E1OBI	UCC2 Ethernet pin option 1 output buffer impedance. This bit controls the output buffer impedance of 'UCC2 pin option 1' output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC2 I/O pins (LVDD1). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts. See Section 3.4.7, "RGMII Pins," for details on reduced pin mode interface.	0
31	UC2E2OBI	UCC2 Ethernet pin option 2 output buffer impedance. This bit controls the output buffer impedance of 'UCC2 pin option 2' output signals, used for reduced pin mode interface (RTBI/RGMII). The output buffer impedance should be correlated to the voltage supplied to the UCC2 I/O pins (LVDD2). 0 Output buffer is set for 40 ohm, 3.3 volts. 1 Output buffer is set for 40 ohm, 2.5 volts. See Section 3.4.7, "RGMII Pins," for details on reduced pin mode interface.	0

5.4.3.7 Debug Configuration

Debug information may be driven on the device pins. This information can identify the internal source of a transaction that reached the (system) DDR SDRAM, secondary DDR SDRAM or local bus interfaces. The device can be configured to drive the MEMC1_MSRCID[0:4], MEMC1_MDVAL,MEMC2_MSRCID[0:4], MEMC2_MDVAL and LSRCID[0:4], and LDVAL signals, respectively on other device pins. The coding of the source ID debug information is the same as the coding of the MSTR_ID field in the AEATR register of the arbiter (See Section 6.2.6, "Arbiter Event Attributes Register (AEATR)").

5.4.3.7.1 DDR Debug Configuration

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

- ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the processor in this mode. Set SICRH[0] to select this mode.
- UART pins. UART operation is disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins. Set SICRL[6–7] to 0b01 to select this mode.

5.4.3.7.2 Secondary DDR Debug Configuration

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

1. ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the secondary SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8360E in this mode. Set SICRH[1] to select this mode.
2. UART pins. UART operation is disabled, and any signals driven by UART devices must be electrically disconnected from the UART I/O pins. Set SICRL[6–7] to 0b10 to select this mode.

5.4.3.7.3 Local Bus Debug Configuration

The local bus debug configuration enables a LBC debug mode in which the SDRAM source ID field and data valid strobe for LBC memory accesses are driven onto UART pins. UART operation must be disabled, and any signals driven by UART devices must be electrically disconnected from the UARTI/O pins in this case. Set SICRL[6–7] to 0b11 to select this mode.

5.4.3.8 DDR Control Driver Register (DDRCDR)

The DDR control driver register (DDRCDR) contains bits that allow control over the driver of the DDR SDRAM controller. Note that the MDIC signals require the use of precision 18-Ω resistors; MDIC0 should be pulled to GND, while MDIC1 should be pulled to GV_{DD}.

The fields in DDRCDR (other than ODT) are used to enable driver calibration with the MDIC[0:1] signals. This can be used to calibrate the DDR drivers to 18 ohms. However, this should only be used for full-strength driver applications. If half strength is desired, then this calibration should remain disabled.

Hardware DDR driver calibration is enabled using DDRCDR[DHC_EN]. If hardware calibration is used, then it should be set before DDR_SDRAM_CFG[MEM_EN] is set.

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is used, the following steps should be taken:

1. Set DDRCDR[DSO_EN] and ensure that DDRCDR[DHC_EN] is cleared.
2. Set the highest impedance (value 0000) for DDRCDR[DSO_PZ].
3. Set DDRCDR[MDIC0_OE] to enable the output enable for MDIC[0].
4. After at least 4 cycles, read DDRDSR[0]. If the value is 0, then use the next lower impedance, and read DDRDSR[0] again. Once a value of 1 is detected, then leave DDRCDR[DSO_PZ] at the calibrated value.
5. Clear DDRCDR[MDIC0_OE].
6. After DDRCDR[DSO_PZ] is calibrated, set a value of 0000 for DDRCDR[DSO_NZ]
7. Set DDRCDR[MDIC1_OE] to enable the output enable for MDIC[1]
8. After at least 4 cycles, read DDRDSR[1]. If the value is 1, then use the next lower impedance, and read DDRDSR[1] again. Once a value of 0 is detected, then leave DDRCDR[DSO_NZ] at the calibrated value

9. Clear DDRCDR[MDIC1_OE]

Note that the legal impedance values, from highest impedance to lowest impedance, are as follows:

- 0000
- 1000
- 1100
- 1110
- 1111

DDRCDR is shown in Figure 5-25.

Offset 0x00128 Access: Read/Write

	0	1	2	5	6	9	10	11	12	13	14	15
R W	DHC_EN	DSO_EN	DSO_PZ		DSO_NZ		MDIC0_OE	MDIC0_OE	ODT	DDR_TYPE	—	
Reset	0	1	1 1	0 0	1 1	0 0	0	0	0	1	0	0

	16	30	31
R W	—		DRQ
Reset	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0	1

Figure 5-25. DDR Control Driver Register (DDRCDR)

Table 5-43 shows the bit definition of the DDRCDR.

Table 5-43. DDRCDR Field Descriptions

Bits	Name	Description
0	DHC_EN	DDR driver hardware compensation enable
1	DSO_EN	0 DDR driver software override disable 1 DDR driver software override enable
2–5	DSO_PZ	DDR driver software p-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal
6–9	DSO_NZ	DDR driver software n-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal
10	MDIC0_OE	Output enable control for MDIC0 pin when software impedance calibration is performed. 0 - MDIC0 is disabled 1 - MDIC0 is enabled
11	MDIC1_OE	Output enable control for MDIC1 pin when software impedance calibration is performed. 0 - MDIC1 is disabled 1 - MDIC1 is enabled

Table 5-43. DDRCDR Field Descriptions (continued)

Bits	Name	Description
12	ODT	ODT termination value for I/Os 0 75 Ohm 1 150 Ohm
13	DDR_TYPE	Selects voltage level for DDR pads 0 DDR2 (1.8V mode) nominal impedance-18 Ohm 1 DDR1 (2.5V mode) nominal impedance-18 Ohm
14–30	—	Reserved
31	DRQ	0 Drain queue before sleep disable 1 Drain queue before sleep enable

5.4.3.9 DDR Debug Status Register (DDRDSR)

Figure 5-26 contains the debug status bits from the DDR SDRAM controller.

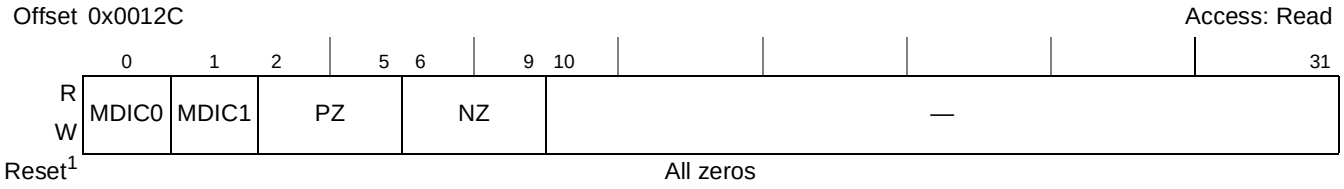


Figure 5-26. DDR Debug Status Register (DDRDSR)

[1] Reset value of bits 0-9 depends on the actual state of the signals being monitored.

Table 5-44 shows the bit settings of the DDRDSR.

Table 5-44. DDRDSR Field Descriptions

Bits	Name	Description
0	MDIC0	DDR driver compensation input value read back for MDIC0. Note that MDIC0 is the PFET driver impedance calibration pin.
1	MDIC1	DDR driver compensation input value read back for MDIC1. Note that MDIC1 is the NFET driver impedance calibration pin.
2–5	PZ	Current setting of PFET driver impedance 0000 Half strength—highest Z 1000 Higher Z than nominal 1100 Nominal impedance setting 1110 Lower Z than nominal 1111 Much lower Z than nominal
6–9	NZ	Current setting of NFET driver impedance 0000 Half strength—highest Z 1000 Higher Z than nominal 1100 Nominal impedance setting 1110 Lower Z than nominal 1111 Much lower Z than nominal
10–31	—	Reserved

5.5 Software Watchdog Timer (WDT)

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

5.5.1 Overview

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

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

Figure 5-27 shows a high-level block diagram of the WDT.

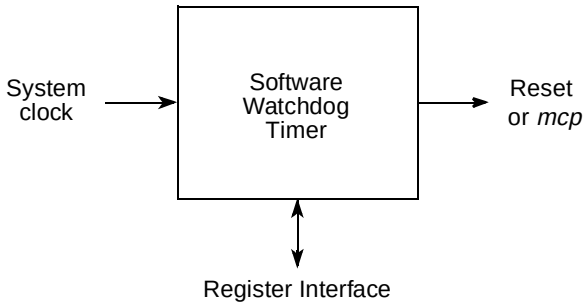


Figure 5-27. Software Watchdog Timer High-Level Block Diagram

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

5.5.2 Features

The WDT includes the following key features:

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

5.5.3 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

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

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

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

5.5.4 Memory Map/Register Definition

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

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

Table 5-45 shows the WDT memory map.

Table 5-45. WDT Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x0–0x3	Reserved	—	—	—
0x4	System watchdog control register (SWCRR)	R/W	0xFFFF_0003 or 0xFFFF_0007[1]	5.5.4.1/5-38
0x8	System watchdog count register (SWCNR)	R	0x0000_FFFF	5.5.4.2/5-38
0xC–0xD	Reserved	—	—	—
0xE	System watchdog service register (SWSRR)	R/W	0x0000	5.5.4.3/5-39

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

5.5.4.1 System Watchdog Control Register (SWCRR)

The system watchdog control register (SWCRR), shown in Figure 5-28, controls the software watchdog period and configures watchdog timer operation. SWCRR can be read at any time but can be written only once after system reset.

Offset 0x4 Access: Read/Write

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

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

Figure 5-28. System Watchdog Control Register (SWCRR)

Table 5-46 defines the bit fields of SWCRR.

Table 5-46. SWCRR Bit Settings

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

5.5.4.2 System Watchdog Count Register (SWCNR)

The system watchdog count register (SWCNR), shown in Figure 5-29, provides visibility to the watchdog counter value. SWCNR is a read-only register. Writes to SWCNR have no effect and terminate without transfer error exception.

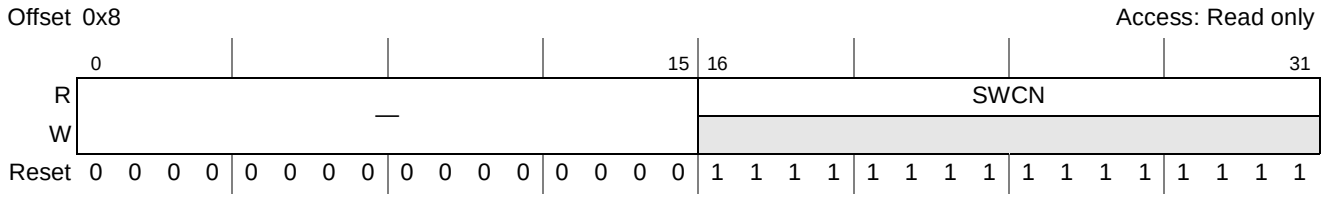


Figure 5-29. System Watchdog Count Register (SWCNR)

Table 5-47 defines the bit fields of SWCNR.

Table 5-47. SWCNR Bit Settings

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

5.5.4.3 System Watchdog Service Register (SWSRR)

The system watchdog service register (SWSRR) is shown in Figure 5-30. When the watchdog timer is enabled, a write of 0x556C followed by a write 0xAA39 to the SWSRR register before the watchdog counter times out prevents a device reset. If the SWSRR register is not serviced before the timeout, a signal from the watchdog timer to the reset or interrupt controller module asserts a system reset or interrupt (depending on the setting of SWCRR[SWRI]).

Both writes must occur before the timeout in the order listed, but any number of instructions can be executed between the two writes. However, writing any value other than 0x556C or 0xAA39 to the SWSRR register resets the servicing sequence, requiring both values to be written to keep the watchdog timer from causing a reset. Reset initializes the SWSRR[WS] field to 0x0000. SWSRR can be written at any time, but returns all zeros when read.

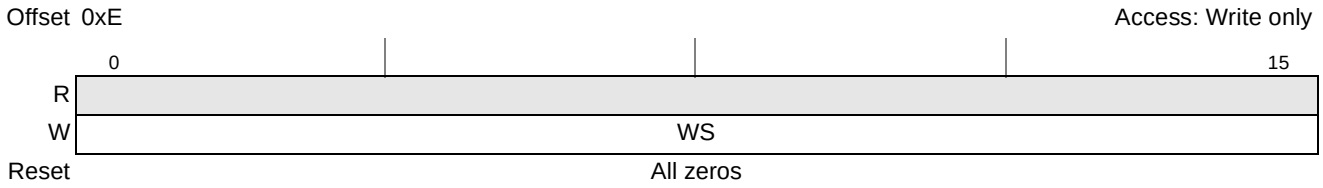


Figure 5-30. System Watchdog Service Register (SWSRR)

Table 5-48 defines the bit fields of SWCNR.

Table 5-48. SWSRR Bit Settings

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

5.5.5 Functional Description

5.5.5.1 Software Watchdog Timer Unit

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

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

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

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

The service sequence reloads the watchdog timer and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSRR, the entire sequence must start over. Although the writes must occur in the correct order before a time-out, any number of instructions can be executed between the writes. This allows interrupts and exceptions to occur between the two writes when necessary. Figure 5-31 shows a state diagram for the watchdog timer.

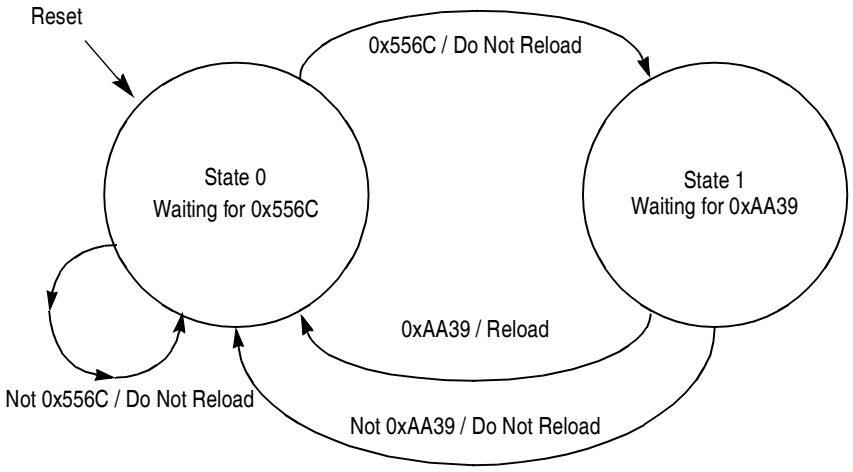


Figure 5-31. Software Watchdog Timer Service State Diagram

Although most software disciplines permit or even encourage the watchdog concept, some systems require a selection of time-out periods. For this reason, the software watchdog timer must provide a selectable range for the time-out period. Figure 5-32 shows how to handle this need.

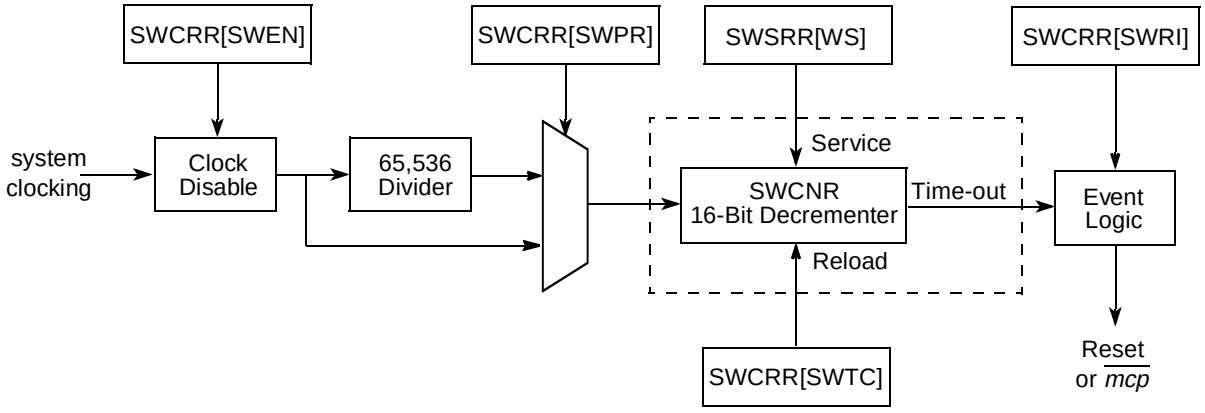


Figure 5-32. Software Watchdog Timer Functional Block Diagram

Figure 5-32 shows that the range is determined by SWCRR[SWTC]. The value in SWTC is then loaded into a 16-bit 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.

5.5.5.2 Modes of Operation

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

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

  — WDT enable mode (SWCRR[SWEN] = 1)

  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.

5.5.6 Initialization/Application Information

5.5.6.1 WDT Programming Guidelines

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

- WDT disabling

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

- WDT initial servicing

  If the software watchdog timer is to be used, the special service sequence, described in Section 5.5.5.1, “Software Watchdog Timer Unit,” must be executed after system reset and not later than the first WDT time-out (~12.8 sec. for 333 MHz system clock).

  Subsequently, periodical WDT servicing should be performed according to the programming guidelines given in Section 5.5.5.1, “Software Watchdog Timer Unit.”

5.6 Real Time Clock Module (RTC)

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

5.6.1 Overview

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

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

Figure 5-33 shows the high level RTC block diagram.

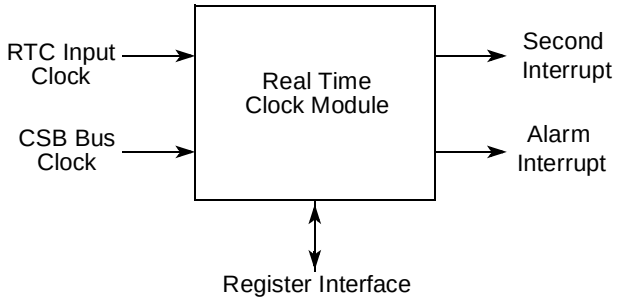


Figure 5-33. Real Time Clock Module High Level Block Diagram

5.6.2 Features

The key features of the RTC include the following:

- Maintains a one-second count, unique over a period of thousand of years.
- 32-bit RTC counter can be initialized by software to specific initial count value.
- Provides an alarm function with programmable and maskable alarm interrupt.
- Provides programmable and maskable every second interrupt.
- Uses two possible clock sources: the CSB bus clock or an external RTC clock.
- RTC function can be disabled if needed.

5.6.3 Modes of Operation

The RTC unit can operate in the following modes:

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

5.6.4 External Signal Description

The following sections provide an overview and detailed descriptions of the RTC signals.

5.6.4.1 Overview

There is one distinct external RTC clock input signal, defined in Table 5-49.

Table 5-49. RTC Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
RTC_CLK	RTC_CLK	Real time clock input. In the MPC8360E, RTC_CLK is one of the possible functions of PC[26]. See Table 3-13 in Chapter 3, "Signal Descriptions."	I	N/A	—

5.6.4.2 Detailed Signal Descriptions

Table 5-50 provides a detailed description of the external RTC signal.

Table 5-50. RTC External Signal—Detailed Signal Description

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

5.6.5 Memory Map/Register Definition

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

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

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

Table 5-45 shows the memory map of the RTC.

Table 5-51. RTC Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Real time counter control register (RTCNR)	R/W	0x0000_0000	5.6.5.1/5-45
0x04	Real time counter load register (RTLDR)	R/W	0x0000_0000	5.6.5.2/5-45
0x08	Real time counter prescale register (RTPSR)	R/W	0x0000_0000	5.6.5.3/5-46
0x0C	Real time counter register (RTCTR)	R	0x0000_0000	5.6.5.4/5-46
0x10	Real time counter event register (RTEVR)	w1c	0x0000_0000	5.6.5.5/5-47
0x14	Real time counter alarm register (RTALR)	R/W	0xFFFF_FFFF	5.6.5.6/5-47
0x18–0x1F	Reserved	—	—	

5.6.5.1 Real Time Counter Control Register (RTCNR)

The real time counter control register (RTCNR), shown in Figure 5-34, is used to enable RTC functions. The register can be read at any time.

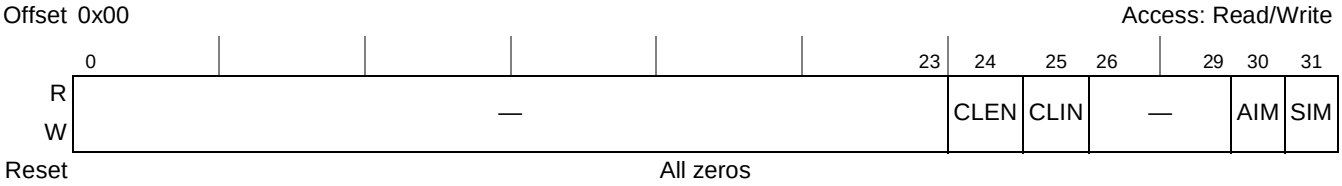


Figure 5-34. Real Time Counter Control Register (RTCNR)

Table 5-52 defines the bit fields of RTCNR.

Table 5-52. RTCNR Bit Settings

Bits	Name	Description
0–23	—	Write reserved, read = 0
24	CLEN	Clock enable control bit. This bit controls the counting of the RTC. When the RTC's clock is disabled, the counter maintains its old value. When the counter's clock is enabled, it continues counting using the previous value. 0 Disable counter. 1 Enable counter.
25	CLIN	Input clock control bit. The input clock to the RTC may be either the CSB clock or an external RTC clock. 0 The input clock to the periodic interrupt timer is CSB input clock. 1 The input clock to the periodic interrupt timer is the external RTC clock.
26–29	—	Write reserved, read = 0
30	AIM	Alarm interrupt mask bit. Used to enable or disable (mask) the RTC alarm interrupt when the RTC's 32-bit counter reaches RTALR[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.

5.6.5.2 Real Time Counter Load Register (RTLDR)

The real time counter load register (RTLDR), shown in Figure 5-35, contains the 32-bit value to be loaded in the 32-bit RTC counter.

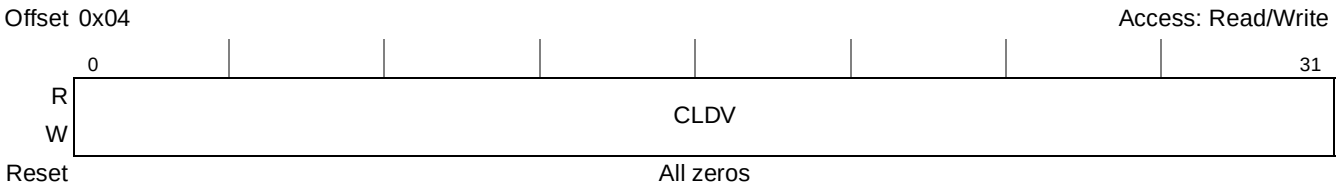


Figure 5-35. Real Time Counter Load Register (RTLDR)

Table 5-53 defines the bit fields of RTLDR.

Table 5-53. RTLDR Bit Settings

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

5.6.5.3 Real Time Counter Prescale Register (RTPSR)

The real time counter prescale register (RTPSR), shown in Figure 5-36, is a read/write register used to configure the RTC prescaler's value.

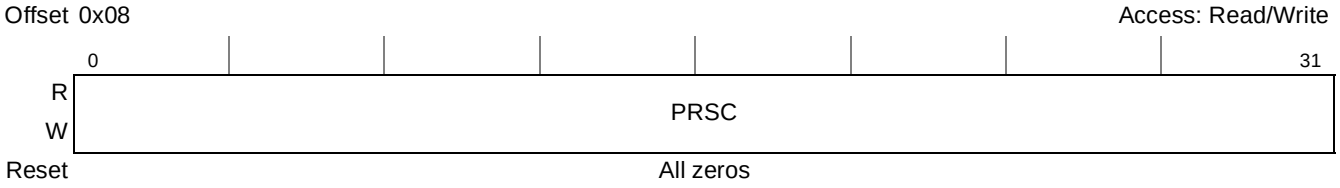


Figure 5-36. Real Time Counter Prescale Register (RTPSR)

Table 5-54 defines the bit fields of RTPSR.

Table 5-54. RTPSR Bit Settings

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

5.6.5.4 Real Time Counter Register (RTCTR)

The real time counter register (RTCTR), shown in Figure 5-37, is a read-only register that shows the current value in the RTC counter.

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

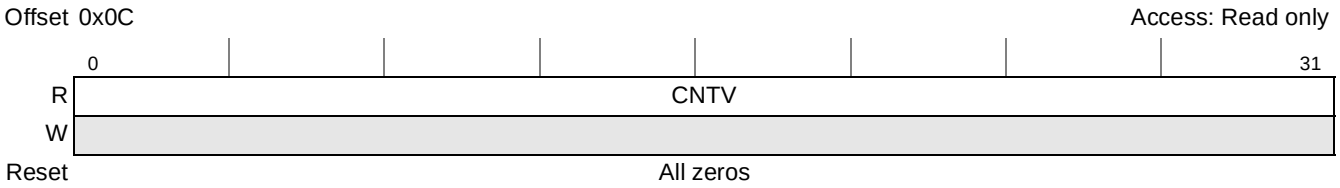


Figure 5-37. Real Time Counter Register (RTCTR)

Table 5-55 defines the bit fields of RTCTR.

Table 5-55. RTCTR Bit Settings

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

5.6.5.5 Real Time Counter Event Register (RTEVR)

The real time counter event register (RTEVR), shown in Figure 5-38, is used to report the source of the interrupts. The register can be read at any time.

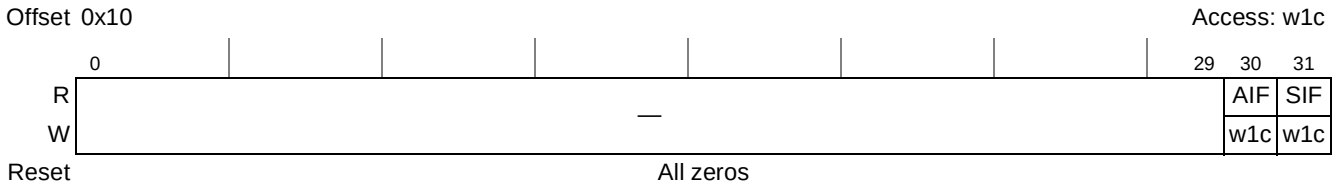


Figure 5-38. Real Time Counter Event Register (RTEVR)

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

Table 5-56 defines the bit fields of RTEVR.

Table 5-56. RTEVR Bit Settings

Bits	Name	Description
0–29	—	Write reserved, read = 0
30	AIF	Alarm interrupt flag bit. Used to indicate the alarm interrupt. 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.

5.6.5.6 Real Time Counter Alarm Register (RTALR)

The real time counter alarm register (RTALR), shown in Figure 5-39, contains the 32-bit alarm (ALRM) value. When the value of the RTC counter equals the RTALR[ALRM] value, a maskable interrupt is generated.

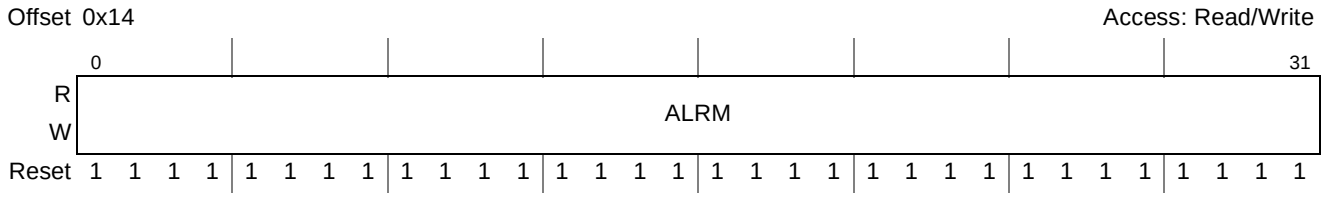


Figure 5-39. Real Time Counter Alarm Register (RTALR)

Table 5-57 defines the bit fields of RTALR.

Table 5-57. RTALR Bit Settings

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

5.6.6 Functional Description

5.6.6.1 Real Time Counter Unit

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

The RTC can be initialized by software with an initial count value in the real time counter load register (RTLDR). It can also be programmed to generate an interrupt every second. The real time counter control register (RTCTR) is used to enable or disable the various timer functions. The real time counter event register (RTEVR) is used to report the interrupt source. The RTC counter is reset to zero on hard reset but is not affected by soft reset. It is initialized by the software. The RTC function can be disabled.

Figure 5-40 shows the functional RTC block diagram.

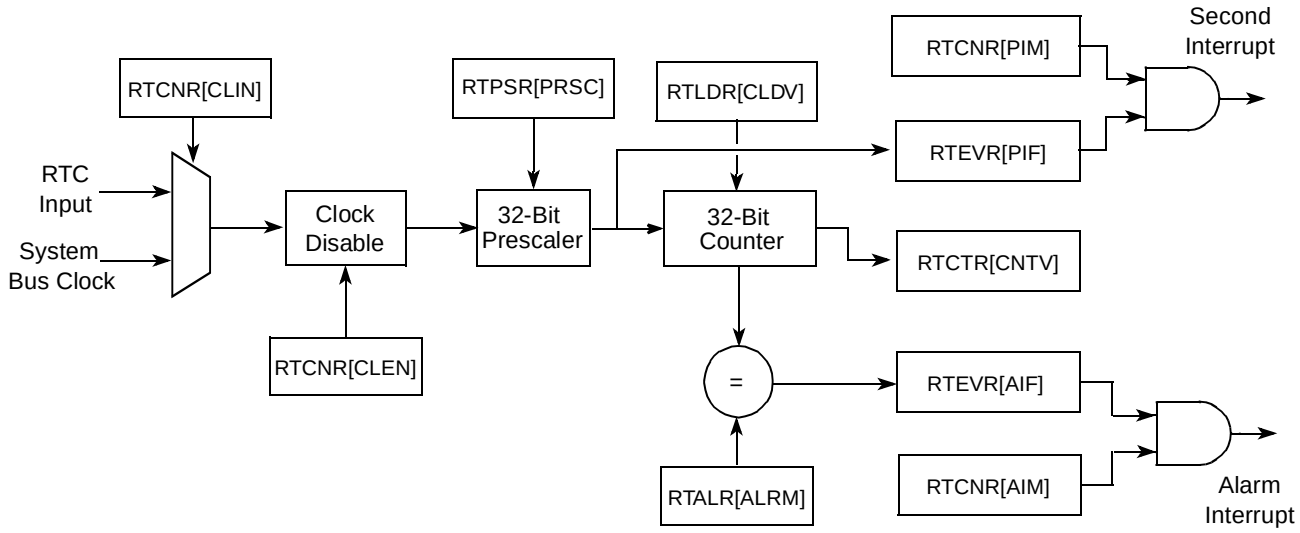


Figure 5-40. Real Time Clock Module Functional Block Diagram

5.6.6.2 RTC Operational Modes

The RTC unit can operate in the following modes:

- RTC enable/disable mode:

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

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

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

  — RTC enable mode (RTCNR[CLEN] = 1)

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

- RTC every-second interrupt enable/disable mode:

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

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

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

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

- RTC alarm interrupt enable/disable mode:

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

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

5.6.7 RTC Programming Guidelines

The following initialization sequence for the RTC is recommended:

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

5.7 Periodic Interval Timer (PIT)

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

5.7.1 Overview

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

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

Figure 5-41 shows the functional PIT block diagram.

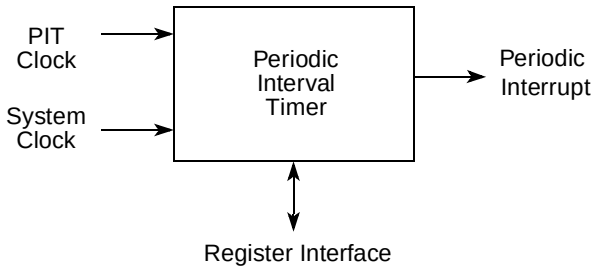


Figure 5-41. Periodic Interval Timer High Level Block Diagram

5.7.2 Features

The key features of the PIT include the following:

- Maintains a 32-bit down-counter, clocked by a 16-bit prescaled input clock
- 32-bit PIT counter can be initialized by software to specific initial count value.
- Provides programmable and maskable periodic interrupt
- Provides maximum period of ~9.5 days (for 333 MHz system clock)
- Uses two possible clock sources: the CSB clock or an external PIT clock
- PIT function can be disabled

5.7.3 Modes of Operation

The PIT unit can operate in the following modes:

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

5.7.4 External Signal Description

The following subsections provide an overview and detailed descriptions of the PIT signals.

5.7.4.1 Overview

There is one distinct external input signal (PIT clock), defined in Table 5-58.

Table 5-58. PIT Signal Properties

Name	Port	Function	I/O	Reset	Pull Up
PIT_CLK	PIT_CLK	Periodic interval timer. In MPC8360E, PIT_CLK is one of the possible functions of PC[26]. See Table 3-13 in Chapter 3, "Signal Descriptions.	I	N/A	—

5.7.4.2 Detailed Signal Description

Table 5-59 describes of the external PIT signal.

Table 5-59. 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	—

5.7.5 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 5-60 shows the PIT memory map.

Table 5-60. PIT Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Periodic interval timer control register (PTCNR)	R/W	0x0000_0000	5.7.5.1/5-52
0x04	Periodic interval timer load register (PTLDR)	R/W	0x0000_0000	5.7.5.2/5-52
0x08	Periodic interval timer prescale register (PTPSR)	R/W	0x0000_0000	5.7.5.3/5-53
0x0C	Periodic interval timer counter register (PTCTR)	R	0x0000_0000	5.7.5.4/5-53
0x10	Periodic interval timer event register (PTEVR)	w1c	0x0000_0000	5.7.5.5/5-54
0x14–0x1F	Reserved	—	—	

5.7.5.1 Periodic Interval Timer Control Register (PTCNR)

The periodic interval timer control register (PTCNR), shown in Figure 5-42, is used to enable the different PIT functions. The register can be read at any time.

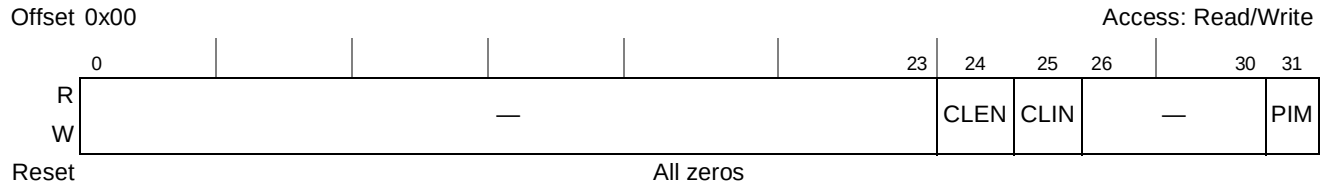


Figure 5-42. Periodic Interval Timer Control Register (PTCNR)

Table 5-61 defines the bit fields of PTCNR.

Table 5-61. PTCNR Bit Settings

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

5.7.5.2 Periodic Interval Timer Load Register (PTLDR)

The periodic interval timer load register (PTLDR), shown in Figure 5-43, contains the 32-bit value to be loaded in a 32-bit PIT counter.

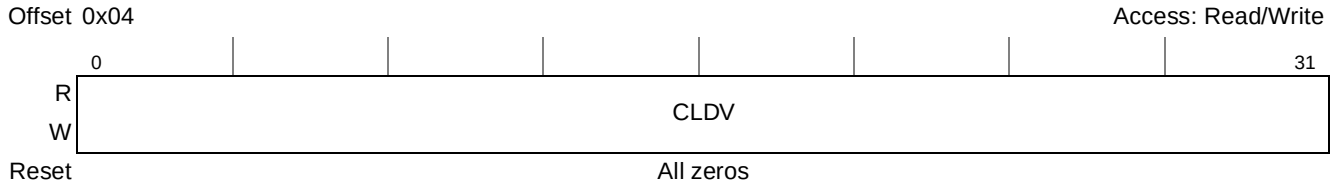


Figure 5-43. Periodic Interval Timer Load Register (PTLDR)

Table 5-62 defines the bit fields of PTLDR.

Table 5-62. PTLDR Bit Settings

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

5.7.5.3 Periodic Interval Timer Prescale Register (PTPSR)

The periodic interval timer prescale register (PTPSR), shown in Figure 5-44, is a read/write register that used to configure the PIT prescaler's value.

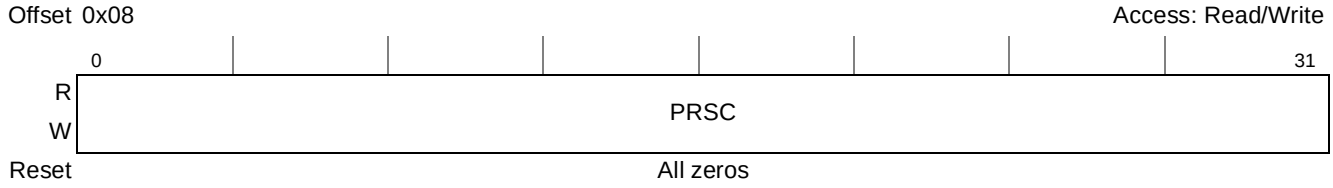


Figure 5-44. Periodic Interval Timer Prescale Register (PTPSR)

Table 5-63 defines the bit fields of PTPSR.

Table 5-63. PTPSR Bit Settings

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

5.7.5.4 Periodic Interval Timer Counter Register (PTCTR)

The periodic interval timer counter register (PTCTR), shown in Figure 5-45, is a read-only register that shows the current value in the PIT counter. The PTCTR counter is not affected by reads or writes.

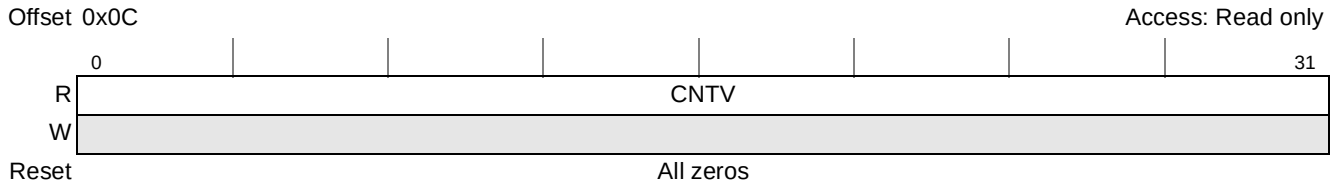


Figure 5-45. Periodic Interval Timer Counter Register (PTCTR)

Table 5-64 defines the bit fields of PTCTR.

Table 5-64. PTCTR Bit Settings

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

5.7.5.5 Periodic Interval Timer Event Register (PTEVR)

The periodic interval timer event register (PTEVR), shown in Figure 5-46, is used to report the source of the interrupts. The register can be read at any time.

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

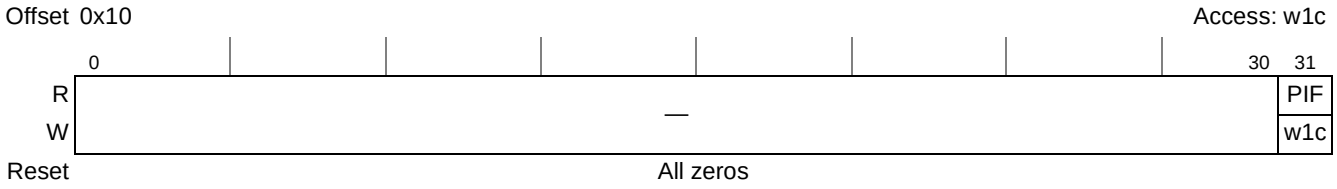


Figure 5-46. Periodic Interval Timer Event Register (PTEVR)

Table 5-65 defines the bit fields of PTEVR.

Table 5-65. PTEVR Bit Settings

Bits	Name	Description
0–30	—	Write reserved, read = 0
31	PIF	Periodic interrupt flag bit. 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.

5.7.6 Functional Description

5.7.6.1 Periodic Interval Timer Unit

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

Figure 5-47 shows the functional PIT block diagram.

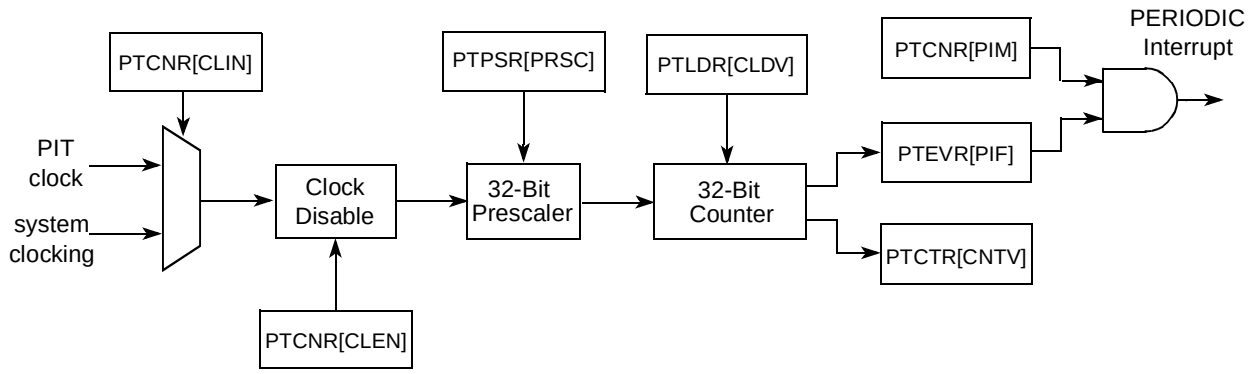


Figure 5-47. Periodic Interval Timer Functional Block Diagram

5.7.6.2 PIT Operational Modes

The PIT unit can operate in the following modes:

- PIT enable/disable mode:

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

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

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

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

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

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

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

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

5.7.7 PIT Programming Guidelines

The following initialization sequence of PIT is recommended:

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

See Section 5.6.7, "RTC Programming Guidelines," for real-time clock programming guidelines.

5.8 General-Purpose Timers (GTMs)

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

5.8.1 Overview

Each global timer module (GTM) includes four identical 16-bit general-purpose timers, two 32-bit timers or one 64-bit timer. Each GTM timer consists of a timer prescale register (GTPSR), a timer mode register (GTMDR), a timer capture register (GTCPR), a timer counter register (GTCNR), a timer reference register (GTRFR), a timer event register (GTEVR), and a timer global configuration register (GTCFR). The GTPSRs and the GTMDRs contain the primary and secondary prescalers, programmed by the user.

Note that while the MPC8360E has two global timer modules, GTM2_TOUT2 and GTM2_TOUT4 signals are not available externally.

Figure 5-48 shows the functional GTM block diagram.

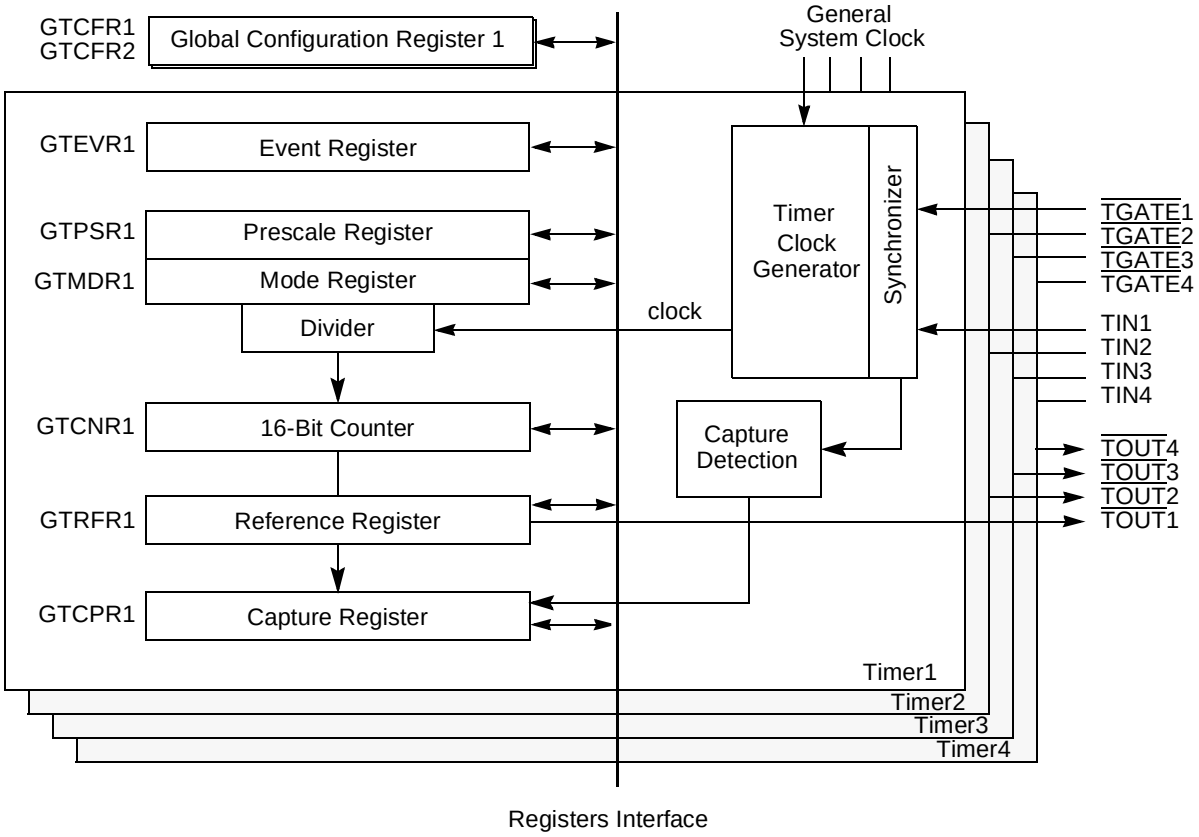


Figure 5-48. Global Timers Block Diagram

5.8.2 Features

The key features of the timer include the following:

- The maximum input clock is the system bus clock
- Four 16-bit programmable timers
- Two timers cascaded internally or externally to form a 32-bit timer
- One timer cascaded internally or externally to form a 64-bit timer
- Maximum period of ~50 msecond (at 333 MHz bus clock) for 16-bit timer
- Maximum period of ~12.8 second (at 333 MHz bus clock) for 32-bit timer
- Maximum period of thousand of year (at 333 MHz bus clock) for 64-bit timer
- 3-nanosecond timer resolution (at 333 MHz bus clock)
- Three programmable input clock sources for the timer prescalers
- Input capture capability
- Output compare with programmable mode for the output pin
- Free run and restart modes
- Functional and programming compatibility with MPC8260 timers

5.8.3 Modes of Operation

The GTM unit can operate in the following modes:

5.8.3.1 Cascaded Modes

GTCFR*n*[PCAS] and 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.

5.8.3.2 Clock Source Modes

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

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

5.8.3.3 Reference Modes

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

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

5.8.3.4 Capture Modes

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

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

5.8.4 External Signal Description

The following sections provide an overview and detailed descriptions of the GTM signals.

5.8.4.1 Overview

There are four distinct external input timer capture signals (TIN1, TIN2, TIN3 and TIN4), four distinct external input timer get signals ($\overline{\text{TGATE}}$1, $\overline{\text{TGATE}}$2, $\overline{\text{TGATE}}$3 and $\overline{\text{TGATE}}$4), and four distinct external timer output signals ($\overline{\text{TOUT}}$1, $\overline{\text{TOUT}}$2, $\overline{\text{TOUT}}$3 and $\overline{\text{TOUT}}$4). The GTM interface signals are defined in Table 5-66.

Table 5-66. GTM Signal Properties

Name	Port	Function	I/O	Reset	Require Pull Up
TIN1	TIN1	Global timer 1 capture control signal	I	0	No
TIN2	TIN2	Global timer 2 capture control signal	I	0	No
TIN3	TIN3	Global timer 3 capture control signal	I	0	No

Table 5-66. GTM Signal Properties (continued)

Name	Port	Function	I/O	Reset	Require Pull Up
TIN4	TIN4	Global timer 4 capture control signal	I	0	No
$\overline{\text{TGATE}}$1	$\overline{\text{TGATE}}$1	Global timer 1 counter gate control signal	I	0	No
$\overline{\text{TGATE}}$2	$\overline{\text{TGATE}}$2	Global timer 2 counter gate control signal	I	0	No
$\overline{\text{TGATE}}$3	$\overline{\text{TGATE}}$3	Global timer 3 counter gate control signal	I	0	No
$\overline{\text{TGATE}}$4	$\overline{\text{TGATE}}$4	Global timer 4 counter gate control signal	I	0	No
$\overline{\text{TOUT}}$1	$\overline{\text{TOUT}}$1	Global timer 1 counter output signal	O	1	No
$\overline{\text{TOUT}}$2	$\overline{\text{TOUT}}$2	Global timer 2 counter output signal	O	1	No
$\overline{\text{TOUT}}$3	$\overline{\text{TOUT}}$3	Global timer 3 counter output signal	O	1	No
$\overline{\text{TOUT}}$4	$\overline{\text{TOUT}}$4	Global timer 4 counter output signal	O	1	No

5.8.4.2 Detailed Signal Descriptions

Table 5-67 provides detailed descriptions of the external GTM signals.

Table 5-67. GTM External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TIN*n*	I	Global timer capture control signal. Used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector.	
		State Meaning	Asserted/Negated —According to the programmed polarity by the corresponding GTMDR*n*[CE]). Each timer has a 16-bit GTCPR used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector. Upon a capture or reference event, the corresponding GTEVR bit is set and a maskable interrupt request is issued to the interrupt controller.
		Timing	Assertion/Negation—Asynchronous to internal bus clock. TIN*n* is internally synchronized to the system bus clock. If TIN*n* meets the asynchronous input setup time, the value of counter is captured after one system bus clock when working with the internal clock.
$\overline{\text{TGATE}}$*n*	I	Global timer counter gate control signal. Used to gate/restart the counter when a defined transition of $\overline{\text{TGATE}}$*n* is sensed by the corresponding input capture edge detector.	
		State Meaning	Asserted/Negated—According to the programmed polarity by the corresponding GTCFR[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.

Table 5-67. GTM External Signals—Detailed Signal Description (continued)

Signal	I/O	Description	
$\overline{\text{TOUT}n}$	O	Global timer counter output signal. The GTM output a signal on the timer output pin $\overline{\text{TOUT}n}$ when the reference value is reached.	
		State Meaning	Asserted/Negated—According to the programmed polarity by the corresponding 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.

5.8.5 Memory Map/Register Definition

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

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

Table 5-68 shows memory map of the GTM.

Table 5-68. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	5.8.5.1/5-61
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	5.8.5.1/5-61
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	5.8.5.2/5-64
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0xFFFF	5.8.5.3/5-66
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R/W	0x0000	5.8.5.4/5-66
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	5.8.5.5/5-67
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	5.8.5.2/5-64
0x22	Timer 4 global timers mode register (GTMDR4)			

Table 5-68. GTM Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0xFFFF	5.8.5.3/5-66
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	5.8.5.4/5-66
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	5.8.5.5/5-67
0x2E	Timer 4 global timers counter register (GTCNR4)			
0x30	Timer 1 global timers event register (GTEVR1)	w1c	0x0000	5.8.5.6/5-67
0x32	Timer 2 global timers event register (GTEVR2)			
0x34	Timer 3 global timers event register (GTEVR3)			
0x36	Timer 4 global timers event register (GTEVR4)			
0x38	Timer 1 global timers prescale register (GTPSR1)	R/W	0x0003	5.8.5.7/5-68
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			

5.8.5.1 Global Timers Configuration Registers (GTCFR*n*)

The global timers configuration registers (GTCFR1 and GTCFR2), shown in Figure 5-49 and Figure 5-50, contain configuration parameters used by the timers. These registers allow simultaneous starting, stopping and resetting of a pair of timers (1 and 2 or 3 and 4) or of a groups of timers (1, 2, 3, and 4) if one bus cycle is used. GTCFR is cleared by reset.

NOTE

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

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 5-49. Global Timers Configuration Register 1 (GTCFR1)

Table 5-69 defines the bit fields of GTCFR1.

Table 5-69. GTCFR1 Bit Settings

Bits	Name	Description
0	PCAS	Pair-cascade mode 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer. **Note:** This bit is ignored in super-cascade mode (GTCFR2[SCAS]=1). **Note:** It is allowed to change the value of this bit only when the corresponding timers are in reset mode. Thus, the user should first clear the RST1 and RST2 bits (without changing PCAS) and then, in a separate write to the register, change the value of PCAS.
1	BCM	Backward compatible mode 0 Provide backward compatibility to PowerQUICC II family timers. In this mode GTCFR1[GM2] bit will control the gate mode for timers 1 and 2 and GTCFR2[GM4] bit will control the gate mode for timers 3 and 4. GTCFR1[GM1] and GTCFR2[GM3] bits are ignored. 1 Normal operational mode
2	STP2	Stop timer 2 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 2, except the Register Interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	RST2	Reset timer 2 0 Reset the timer 2, including GTMDR2, GTRFR2, GTCNR2, GTCPR2 and GTEVR2 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP2 bit is cleared.
4	GM2	Gate mode for $\overline{\text{TGATE2}}$ 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 5-50.

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 5-50. Global Timers Configuration Register 2 (GTCFR2)

Table 5-70 defines the bit fields of GTCFR2.

Table 5-70. GTCFR2 Bit Settings

Bits	Name	Description
0	PCAS	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].

Table 5-70. GTCFR2 Bit Settings (continued)

Bits	Name	Description
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 3and 4.
6	STP3	Stop timer 3 0 Normal operation 1 Reduce power consumption of the corresponding timer. This bit stops all clocks to the timer 3, except the register interface clock, which allows to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	RST3	Reset timer 3 0 Reset the timer 3, including GTMDR3, GTRFR3, GTCNR3, GTCPR3 and GTEVR3 (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP3 bit is cleared.

5.8.5.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

The global timers mode registers (GTMDR1, GTMDR2, GTMDR3 and GTMDR4) are shown in Figure 5-51.

Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDR*n*. Only GTCFR*n*[RST*n*] and GTCFR*n*[STP*n*] can be modified at any time.

Offset 0x10(GTMDR1) 0x20(GTMDR3) Access: Read/Write
0x12(GTMDR2) 0x22(GTMDR4)

	0	7	8	9	10	11	12	13	14	15
R W	SPS		CE		OM	ORI	FRR	ICLK		GE
Reset	All zeros									

Figure 5-51. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 5-71 defines the bit fields of GTMDR.

Table 5-71. GTMDR Bit Settings

Bits	Name	Description
0–7	SPS	Secondary prescaler value The secondary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.
8–9	CE	Capture edge and enable interrupt 00 Disable interrupt on capture event; capture function is disabled 01 Capture on rising TIN*n* edge only and enable interrupt on capture event. 10 Capture on falling TIN*n* edge only and enable interrupt on capture event. 11 Capture on any TIN*n* edge and enable interrupt on capture event. **Note:** The frequency of TIN*n* should be slower than system clock (TIN*n* is sampled internally by system clock to detect TIN*n*'s rising/falling edge before updating the counter)
10	OM	Output mode 0 Toggle $\overline{\text{TOUT}}$*n* every time when the corresponding timer matches its reference value. 1 Active-low pulse on $\overline{\text{TOUT}}$*n* for one timer input clock cycle (4 input clock cycles for the system clock) as defined by the ICLK*n* bits. Thus, $\overline{\text{TOUT}}$*n* may be low for four general system clocks, one general system slow go clock period, or one TIN*n* pin clock cycle period. **Note:** $\overline{\text{TOUT}}$*n* changes are internally synchronized to the rising edge of the system clock
11	ORI	Output reference interrupt enable 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt 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 general system bus clock. 10 Internal slow go clock (divided by 16 system bus clock). 11 TIN*n*: corresponding TIN1, TIN2, TIN3 or TIN4 pin (falling edge).
15	GE	Gate enable 0 The $\overline{\text{TGATE}}$*n* signal is ignored. 1 The $\overline{\text{TGATE}}$*n* signal is used to control the timer.

5.8.5.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

Global timers reference registers, shown in Figure 5-52, are 16-bit memory-mapped, read/write registers containing the 16-bit reference values for each timer's timeout. The reference value is not reached until GTCNR*n*[CNV] increments to the value in GTRFR*n*[TRV].

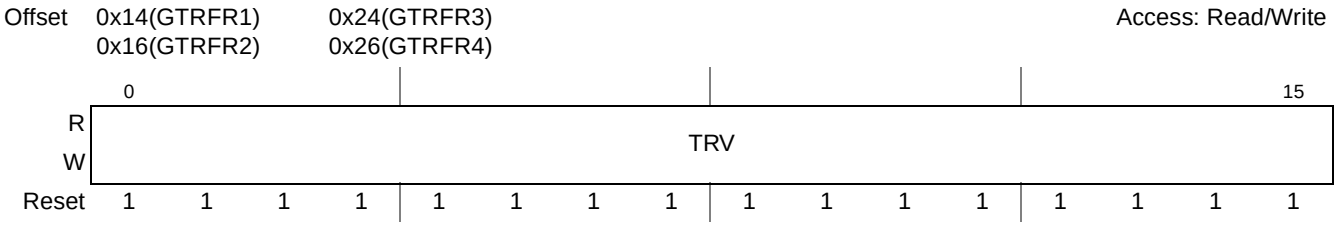


Figure 5-52. Global Timers Reference Registers (GTRFR1–GTRFR4)

Table 5-72 defines the bit fields of GTRFR.

Table 5-72. GTRFR Bit Settings

Bits	Name	Description
0–15	TRV	Timeout reference value. 16-bit timeout reference value for the corresponding timer. Set to all ones by reset.

5.8.5.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

Global timers capture registers (GTCPR1, GTCPR2, GTCPR3 and GTCPR4), shown in Figure 5-53, are used to latch the value of the counters according to GTMDR*n*[CE].

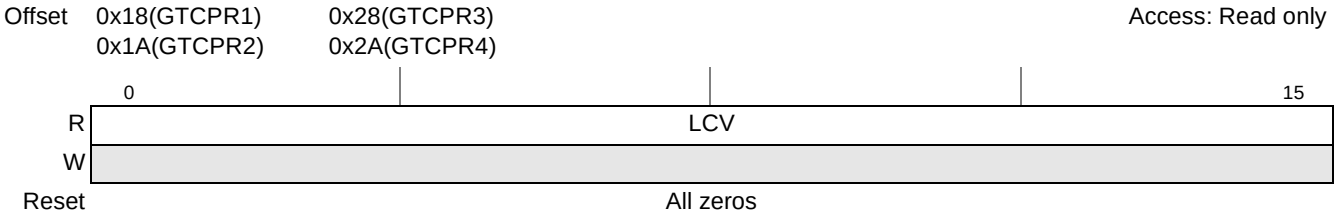


Figure 5-53. Global Timers Capture Registers (GTCPR1–GTCPR4)

Table 5-73 defines the bit fields of GTCPR*n*.

Table 5-73. GTCPR*n* Bit Settings

Bits	Name	Description
0–15	LCV	Latched counter value. Corresponding timer's 16-bit latched value.

5.8.5.5 Global Timers Counter Registers (GTCNR1–GTCNR4)

Global timers counter registers (GTCNR1, GTCNR2, GTCNR3 and GTCNR4), shown in Figure 5-54, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a GTCNR*n*[CNV] fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a GTCNR*n*[CNV] field sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

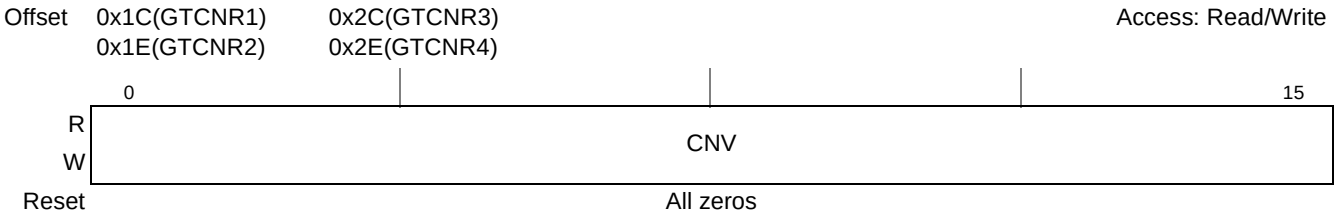


Figure 5-54. Global Timers Counter Registers (GTCNR1—GTCNR4)

Table 5-74 defines the bit fields of GTCNR.

Table 5-74. GTCNR Bit Settings

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

5.8.5.6 Global Timers Event Registers (GTEVR1–GTEVR4)

Global timers event registers (GTEVR1, GTEVR2, GTEVR3 and GTEVR4), shown in Figure 5-55, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets GTEVR*n*[REF], regardless of the corresponding GTMDR*n*[ORI]. The capture event is only set if it is enabled by GTMDR*n*[CE]. GTEVRs appear as memory-mapped registers to users, which can be read at any time.

GTEVR*n* bits are cleared by writing ones to them (writing zeros does not affect bit values). Both bits must be reset before the timer negates the interrupt to the interrupt controller.

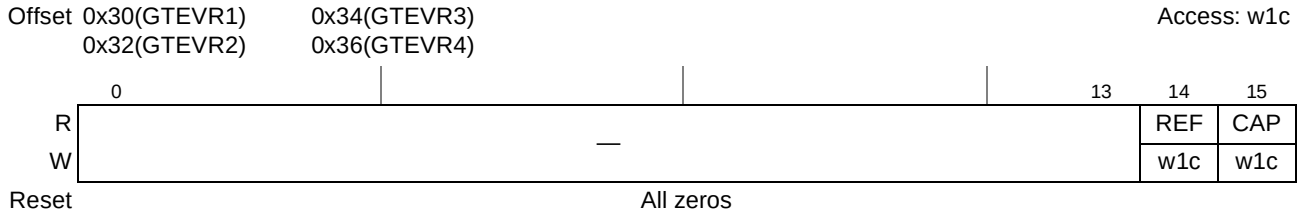


Figure 5-55. Global Timers Event Registers (GTEVR1—GTEVR4)

Table 5-75 defines the bit fields of GTEVR*n*.

Table 5-75. GTEVR*n* Bit Settings

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event 0 No event 1 The counter reached the GTRFR*n*[TRV] value. GTMDR*n*[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event Corresponding timer's 16-bit read/write up-counter value. 0 No event 1 The counter value has been latched into the GTCPR*n*[LCV]. GTMDR*n*[CE] is used to enable generation of this event.

5.8.5.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

The global timers prescale registers (GTPSR1, GTPSR2, GTPSR3 and GTPSR4) are shown in Figure 5-56.

Erratic behavior may occur if GTPSR*n* is not initialized before the corresponding GTMDR*n*.

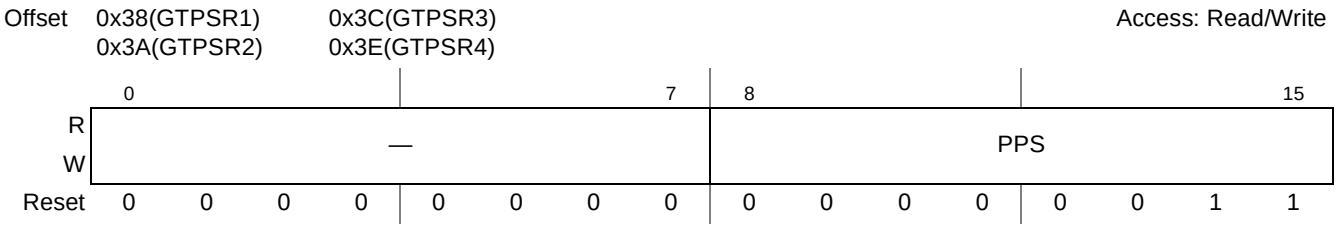


Figure 5-56. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 5-76 defines the bit fields of GTPSR*n*.

Table 5-76. GTPSR*n* Bit Settings

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	PPS	Primary prescaler bits The primary prescaler is programmed to divide the clock input to corresponding timer by values from 1 to 256. The value 0x00 divides the clock by 1 and 0xFF divides the clock by 256.

NOTE

The total timer prescale value is calculated as follows:

$$GTMn_{prescaler} = (GTPSRn[PPS] + 1) \cdot (GTMDRn[SPS] + 1)$$

This gives a total prescale range from 1 (GTPSR*n*[PPS] = 0x00, GTMDR*n*[SPS] = 0x00) to 65,536 (GTPSR*n*[PPS] = 0xFF, GTMDR[SPS] = 0xFF).

5.8.6 Functional Description

5.8.6.1 General-Purpose Timer Units

The clock input to the timer's prescaler can be selected from the following sources:

- The system clock
- The system slow go clock (internally divided by 16)

The general system clock is generated in the clock synthesizer and defaults to the system frequency. However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the general system clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer TIN*n* to be the clock source. TIN*n* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding GTMDR*n*[ICLK] bits. The prescalers (GTMDR*n*[SPS] and GTPSR*n*[PPS]) can be programmed to divide the clock input by values from 1 to 65,537 and the output of the prescaler is used as an input to the 16-bit counters. The best resolution of the timer is one clock cycle (3 ns at a 333-MHz system clock, for example). The maximum period (when the reference value is all ones) for one 16-bit timer is ~50 ms at 333-MHz.

5.8.6.2 Reference Modes

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

- Free run reference mode (GTMDR*n*[FRR] = 0)

  The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode (GTMDR*n*[FRR] = 1)

  The corresponding timer count is reset immediately after the reference value is reached.

Upon reaching the reference value, the corresponding GTEVR*n*[REF] bit is set and an interrupt is issued if GTMDR*n*[ORI] = 1. The timers can output a signal on the timer output pin $\overline{\text{TOUT}}$*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.

5.8.6.3 Capture Modes

In addition, each timer has a 16-bit field in GTCPR, used to latch the value of the counter when a defined transition of TIN*n* is sensed by the corresponding input capture edge detector. The timers may be gated/restarted by an external gate signals ($\overline{\text{TGATE}}$*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.

5.8.6.4 Cascaded Modes

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

- Non-cascaded mode (GTCFR*n*[PCAS] = 0 and GTGCF2[SCAS] = 0)

  If GTCFR*n*[PCAS] = 0 and GTCFR2[SCAS] = 0, the each timer (timer 1, timer 2, timer 3 and timer 4), function as a independent 16-bit timer with a 16-bit GTRFR, GTCPR, GTMDR and GTCNR for each one (Figure 5-57). When working in the none-cascaded mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with appropriate 16-bit bus cycles.

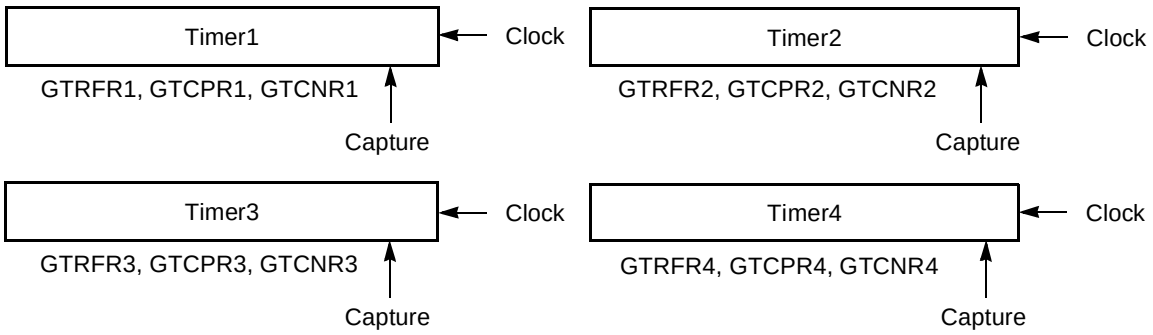


Figure 5-57. Timers Non-Cascaded Mode Block Diagram

- Pair-cascaded mode (GTCFR1[PCAS] = 1 and/or GTCFR2[PCAS] = 1, GTCFR2[SCAS] = 0)

  In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter: timer 1 may be internally cascaded to timer 2 and timer 3 may be internally cascaded to timer 4, as shown in Figure 5-58. Since the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer (GTCFR1[PCAS] = 1, GTCFR2[PCAS] = 0 or GTCFR1[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 timer4) function as a 32-bit timer with a 32-bit GTRFR, GTCPR, and GTCNR. In this case, GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4 and GTCFR1/GTCFR2. The capture are controlled from TIN2, and the interrupts are generated from GTEVR2. When working in the pair-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with 32-bit bus cycles.

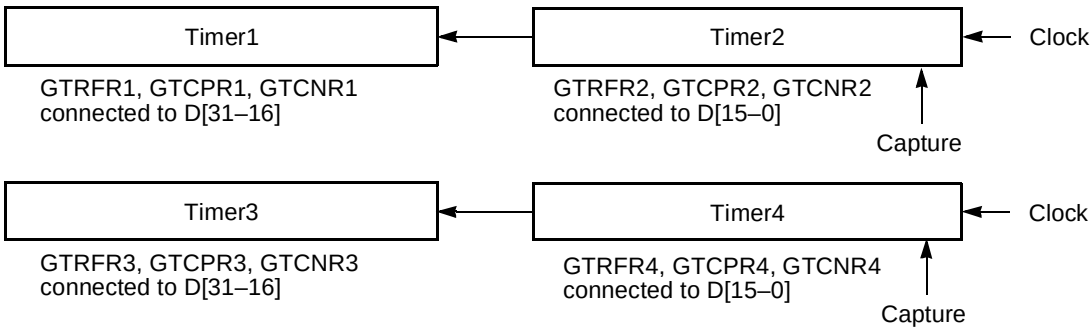


Figure 5-58. Timer Pair-Cascaded Mode Block Diagram

- Super-cascaded mode (GTCFR2[SCAS] = 1)

  In this mode, all four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in Figure 5-59.

  If GTCFR2[SCAS] = 1, the all four 16-bit timers function as a 64-bit timer with a cascaded 32-bit GTRFR, GTCPR, and GTCNR. In this case, registers GTMDR1, GTMDR2 GTMDR3 and GTCFR1 are ignored, and the modes and functions are defined using GTMDR4 and GTCFR2 only. The capture are controlled from TIN4, and the interrupts are generated from GTEVR4. When working in the super-cascaded mode, the cascaded GTRFR, GTCPR, and GTCNR should be referenced with two 32-bit bus cycles.

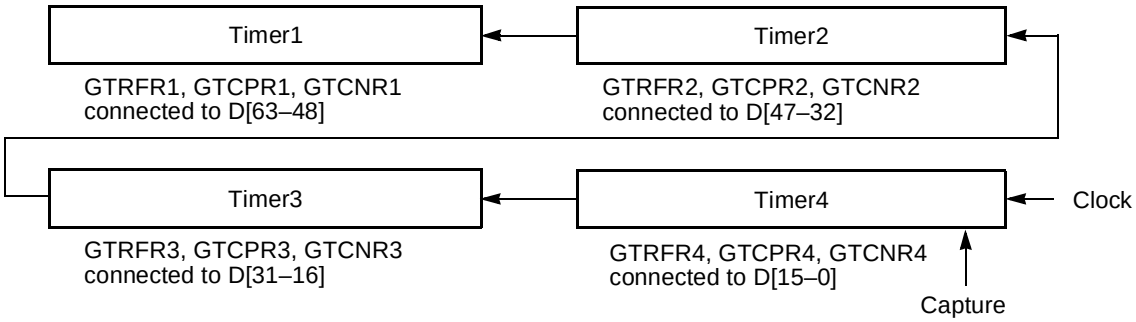


Figure 5-59. Timers Super-Cascaded Mode Block Diagram

5.8.7 Initialization/Application Information

5.8.7.1 Programming Guidelines

5.8.7.1.1 GTM Registers

The following initialization sequence of GTM is recommended:

- Write to GTCFR*n* in order to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
- Write to GTPSR*n*[PPS] fields in order to program the appropriate timer's clock primary prescaler.
- Write to GTMDR*n* in order to choose an input clock, to program the secondary prescaler and to set a desirable appropriate timer's operational mode.

NOTE

Erratic behavior may occur if GTCFR*n* and GTPSR are not initialized before the GTMDR. Only GTCFR*n*[RST*n*] can be modified at any time

- Clear GTEVR*n*[REF] and GTEVR*n*[CAP] by writing 1's in order to clear the previous events.
- Write to GTRFR and to GTCNR*n* according to appropriate timer's GTMDR*n* programming.

NOTE

A write cycle to a GTCNR*n*[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescalers, (GTPSR*n*[PPS] and GTMDR*n*[SPS]), to be reset.

- Write to GTCFR*n*[STP*n*] and to GTCFR*n*[RST*n*] in order to initialize the appropriate timer's operation.

5.9 Power Management Control

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.

5.9.1 Modes of Operation

The device uses one or more of the following methods for power saving:

- Dynamic power management
- Shutting down unused blocks
- Software-controlled power-down states

5.9.2 External Signal Description

Table 5-77 describes the power management signals.

Table 5-77. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{QUIESCE}}$	O	Quiesce state. Indicates that the processor system and PowerPC core are in low power state.	
		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.

5.9.3 Memory Map/Register Definition

Table 5-78 shows the memory map for the power management controller registers.

Table 5-78. Power Management Controller Registers Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x00B00	Power management controller configuration register (PMCCR)	R/W	0x0000_0000	5.9.3.1/5-73
0x00B04	Power management controller event register (PMCER)	R/W	0x0000_0000	5.9.3.2/5-74
0x00B08	Power management controller mask register (PMCMR)	R/W	0x0000_0000	5.9.3.3/5-75
0x00B0C–0x00BFC	Reserved	—	—	—

5.9.3.1 Power Management Controller Configuration Register (PMCCR)

The power management controller configuration register (PMCCR), shown in Figure 5-60, controls whether only the PowerPC core will enter low power state upon quiesce request or additional parts of the device will also enter low power state.

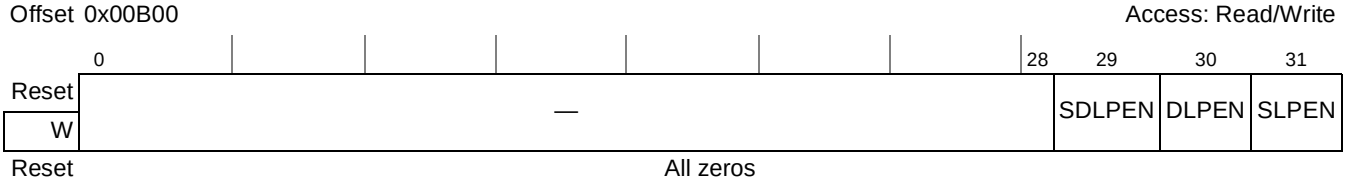


Figure 5-60. Power Management Controller Configuration Register

Table 5-5 defines the bit fields of PMCCR.

Table 5-79. PMCCR Bit Settings

Bits	Name	Description
0–28	—	Reserved. Write has no effect, read returns 0.
29	SDLPEN	Secondary DDR SDRAM low power enable 0 The secondary DDR SDRAM memory controller is prevented from entering low power state. 1 The secondary DDR SDRAM memory controller will enter low power state when the rest of the system enters low power state, according to SLPEN setting. Secondary DDR SDRAM will enter self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) and clocks (MCK*n*) are shut off. This bit is cleared when the MPC8360E exits from low power state. Note that setting this bit without setting SLPEN has no effect.
30	DLPEN	DDR SDRAM low power enable 0 The DDR SDRAM memory controller is prevented from entering low power state. 1 The DDR SDRAM memory controller will enter low power state when the rest of the system enters low power, according to SLPEN setting. DDR SDRAM will enter self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) and DDR clocks (MCK*n*) are shut off. This bit is cleared when the device exits from low power state. Note that setting this bit without setting SLPEN has no effect.
31	SLPEN	System low power enable 0 The system is prevented from entering low power state. 1 The system will enter low power state when a quiesce request from the PowerPC core arrives. This bit is cleared when the device exits from low power state.

5.9.3.2 Power Management Controller Event Register (PMCER)

The power management controller event register (PMCER), shown in Figure 5-61, indicates with the PMCI bit that the power management controller has detected a wakeup 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 5-61. Power Management Controller Event Register

Table 5-80 defines the bit fields of PMCER.

Table 5-80. PMCER Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	PMCI	Power management controller interrupt. When set, indicates that the power management controller has detected that the system is not in idle state anymore, and that the device is required to exit low power state. If PMCMR[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).

5.9.3.3 Power Management Controller Mask Register (PMCMR)

The power management controller mask register (PMCMR), shown in Figure 5-62, 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.

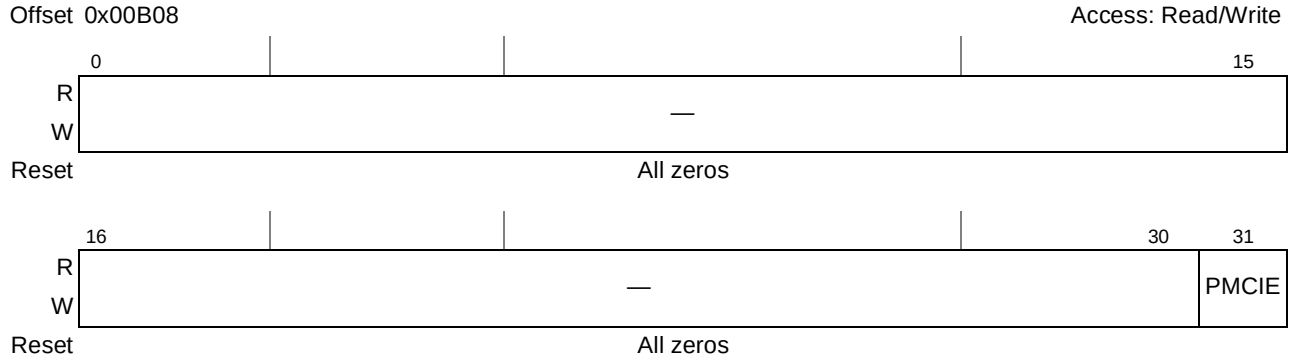


Figure 5-62. Power Management Controller Mask Register

Table 5-81 defines the bit fields of PMCMR.

Table 5-81. PMCMR Bit Settings

Bits	Name	Description
0–30	—	Reserved. Write has no effect, read returns 0.
31	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 SIMR_L[PMC].

5.9.4 Functional Description

The device has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to individual blocks when they are not needed through a memory-mapped register in the clock unit (SCCR).

Additionally, software running on the PowerPC core can access the core's SPRs to put the device into doze, nap, or sleep power down states. Finally, software can access the PMCCR register to enable the device to go to low power state whenever the PowerPC core enters nap or sleep states. These power management features are described in further detail in this section.

5.9.4.1 Dynamic Power Management

Many blocks in the device can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

5.9.4.2 Shutting Down Unused Blocks

As described in Section 4.5.2.3, "System Clock Control Register (SCCR)," SCCR provides a way to shut down certain functional blocks within the device when they are not needed in a particular system. SCCR can be written by the PowerPC core or by an external master. Powering down a block in this way turns off all clocks to that block. It does not remove power. It is required that the SCCR is written to shut down a certain functional block only when that block is idle.

NOTE

Functional blocks disabled using SCCR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

5.9.4.3 Software-Controlled Power-Down States

PowerPC software can place the core in doze, nap, or sleep power-down states by writing to HID0 in the core, as described in detail in the section "Hardware Implementation Register 0 (HID0)," of the *e300 PowerPC Core Reference Manual*. In addition, if PMCCR[SLPEN] is set when the PowerPC core request to enter nap or sleep modes, it will also cause the system internal logic units to enter low power mode.

5.9.4.3.1 Entering Low Power States—Core-Only Mode

Entering Doze mode is controlled only by the e300 PowerPC core itself, and does not involve the power management controller or other blocks. For a more detailed description, see Table 7-1.

Entering Nap or Sleep modes occurs by writing to HID0 in the core, causing the core to make a quiesce request to the power management controller while PMCCR[SLPEN] is cleared. The core is immediately enabled to enter low power state, regardless of the system status. Note that since the core does not snoop the bus in this mode, it is the user's responsibility to keep the cache coherent. Other device peripheral and internal units continue to operate in full-on mode while the core is in low power state in this mode.

5.9.4.3.2 Entering Low Power States—Core and System Mode

Core and system mode is achieved when the core makes a quiesce request to the power management controller after PMCCR[SLPEN] is set. To preserve cache coherency and otherwise avoid loss of system state, the core's transition to low-power modes is coordinated with other functional blocks. The power management controller allows the core to enter power down mode only when the rest of the system is idle.

When the power management controller detects that the internal system bus is idle, and there are no outstanding transactions, it signals the internal logic units to enter low power state.

If PMCCR[DLPEN] and/or PMCCR[SDLPEN] is set, the DDR SDRAM and/or secondary DDR SDRAM is first set to self-refresh mode (if enabled by DDR_SDRAM_CFG[SREN] memory controller register) before the memory controller stops driving refresh commands. Self-refresh mode guarantees that the memory content will remain valid while the memory controller and its clocks are off. The DDR clocks are then disabled. Finally the DDR SDRAM memory controller enters low power state and acknowledges the power management controller.

The power management controller then signals the core and acknowledges it's request to enter power down mode. Finally the $\overline{\text{QUIESCE}}$ output signal is asserted.

5.9.4.4 Exiting Core and System Low Power States

The device can exit low power state and return to full-on mode for one of the following reasons:

- The core internal time base unit invokes a request to exit low power state.
- The core has received an interrupt request.
- The power management controller has detected that the system is not idle and there are outstanding requests for transactions on the internal bus.

The actions taken to exit low power state depend on the mode and whether the system or only the core are in this state. The following sections describe the various scenarios.

5.9.4.4.1 Exiting Low Power States—Core-Only Mode

Exit from Doze mode is controlled only by the core itself and does not involve the power management controller or other blocks in the device. For a detailed description, see Table 7-1.

Nap or Sleep modes are exited when the core has received an interrupt request, or according to the internal time base unit of the core (Nap mode only). The source of the interrupt can be an internal block or external signal. When the core returns to full-on state, it signals to the power management controller that it is ready and is immediately acknowledged to access the rest of the system.

5.9.4.4.2 Exiting Low Power States—Core and System Mode

The power management controller decides to exit low power state when it detects that the system is not idle anymore. The device may exit idle state when one of the peripheral interfaces makes a request to access the internal bus or when the core returns to full-on state, as described above, and makes a request to access the internal bus. For example, the QUICC Engine block receives an Ethernet frame, and requires to store it on the DDR SDRAM memory.

If the particular DDR SDRAM memory controller is in low power state (PMCCR[DxLPEN] was set when entering low power state), the power management controller initially enables the DDR SDRAM memory controller. DDR SDRAM clocks (MCK*n)* are enabled and the memory controller exit self-refresh and returns to auto-refresh mode.

The power management controller then enables other internal units and interrupts the PowerPC core. When all internal units, including the core, are ready, the power management controller enables the device

to return to full-on state, negate the $\overline{\text{QUIESCE}}$ output, and clear 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].

5.9.5 Initialization/Application Information

5.9.5.1 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 5.4.3.4, "System Priority and Configuration Register (SPCR)," for more information.
3. The e300 core enters low power state by accessing the HID0 register.
4. Set 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.

Part III
Core and I/O Interfaces

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

- Chapter 6, "Arbiter and Bus Monitor," provides an overview of the arbiter in the MPC8360E device. It also describes configuration, control, and status registers of the arbiter.
- Chapter 7, "e300 Processor Core Overview," provides an overview of the basic functionality of the e300 processor core and briefly describes how the functional units interact.
- Chapter 8, "Integrated Programmable Interrupt Controller (IPIC)," describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. It also provides a definition of the external interrupt signals and their functions. In addition, the interrupt configuration, control, and status registers are described in this chapter.
- Chapter 9, "DDR Memory Controller," describes the DDR SDRAM memory controller of the MPC8360E. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. Special features like ECC error injection support rapid system debug.
- Chapter 10, "Local Bus Controller," describes the local bus controller (LBC) of the MPC8360E. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Also, it includes an initialization and applications information section with many specific examples of its use.
- Chapter 11, "Sequencer," describes how the I/O sequencer switches transactions among its ports, using a buffer pool to minimize blocking. It also provides address translation on outbound PCI transactions.
- Chapter 12, "DMA/Messaging Unit," describes the four-channel high speed general-purpose DMA controller of the MPC8360E. The channels share buffer space in the IOS to facilitate the gathering and sending of data. The DMA/messaging unit supports communication between two processors on different buses, for example, a local processor and a processor on a PCI bus. This communication unit operates with generic messages and doorbell registers. This block also provides a DMA controller that transfers blocks of data independent of the local processor or PCI hosts.
- Chapter 13, "PCI Bus Interface," describes the PCI interface, which complies with the *PCI Local Bus Specification*, Rev. 2.3. This chapter provides a basic description of PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification.

- Chapter 14, “Security Engine (SEC) 2.4,” describes the SEC 2.4 that is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the e300 core of the MPC8360E. It is optimized to process all the algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and IEEE Std .802.11i.™ .
- Chapter 15, “I^2C Interfaces,” describes the inter-IC (IIC or I^2C) bus controller of the MPC8360E. This synchronous, serial, bidirectional, multiple-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8360E powers up in boot sequencer mode which allows the 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 MPC8360E to facilitate boundary-scan testing. The JTAG interface complies with the IEEE Std. 1149.1™ boundary-scan specification.
- Chapter 18, “Delay Lock Loop (DLL),” describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described.

Chapter 6
Arbiter and Bus Monitor

This chapter describes operation theory of the arbiter in the device. In addition, it describes configuration, control, and status registers of the arbiter.

6.1 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.

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	0xFFFF_FFFF	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.

Offset 0x00 Access: User read/write

	0	6	7	8	12	13	15
R	—		COREDIS	—		PIPE_DEP	
W							
Reset	All zeros[1]						

	16	17	19	20	21	23	24	25	26	27	28	31
R	—	PCI_RPTCNT		—	RPTCNT		—		APARK		PARKM	
W												
Reset	All zeros[1]											

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 it's value from the reset configuration bit of COREDIS and can be configured by software. Also, if boot source is boot sequencer, COREDIS must be set to 1 at reset and the last transaction of the boot sequencer must set COREDIS to 0, if CPU enable is needed. 0 CPU enabled. 1 CPU disabled.
8–9	—	Write reserved, read = 0
10–11	—	Reserved. Write should preserve reset value. The reset value is a function of the core PLL configuration, which is part of the reset configuration word. When the core is set to operate at 1:1 or 3:2 bus clock, these bits are set to '01' during reset; otherwise, they are set to '00'.
12	—	Write reserved, read = 0
13–15	PIPE_DEP	Pipeline depth (number of outstanding transactions). 000 Pipeline depth 1 (1 outstanding transaction) 001 Pipeline depth 2 (2 outstanding transactions) 010 Pipeline depth 3 (3 outstanding transactions) 011 Pipeline depth 4 (4 outstanding transactions) 1xx Reserved
16	—	Write reserved, read = 0
17–19	PCI_RPTCNT	PCI repeat count. Specifies the maximum number of consecutive transactions, that PCI master can perform, using $\overline{\text{REPEAT}}$ request mode. 000 One consecutive transaction ($\overline{\text{REPEAT}}$ request mode disable) 001 Two consecutive transactions 010 Three consecutive transactions 011 Four consecutive transactions 100 Five consecutive transactions 101 Six consecutive transactions 110 Seven consecutive transactions 111 Eight consecutive transactions
20	—	Write reserved, read = 0
21–23	RPTCNT	Repeat count. Specifies the maximum number of consecutive transactions, that any master (except PCI) can perform, using $\overline{\text{REPEAT}}$ request mode. 000 1 consecutive transactions ($\overline{\text{REPEAT}}$ request mode disable) 001 2 consecutive transactions 010 3 consecutive transactions 011 4 consecutive transactions 100 5 consecutive transactions 101 6 consecutive transactions 110 7 consecutive transactions 111 8 consecutive transactions **Note:** It is recommended not to program this field for more than four consecutive transactions.
24–25	—	Write reserved, read = 0

Table 6-2. ACR Field Descriptions (continued)

Bits	Name	Description
26–27	APARK	Address parking. Specifies arbiter bus parking mode. 00 Park to master. Arbiter parks the address bus to the master, that is selected by numeric value of PARKM field. 01 Park to last owner. Arbiter parks the address bus to last bus owner. 10 Disable. Arbiter does not assert $\overline{BG}$ to any master, if no $\overline{BR}$ is present. 11 Reserved
28–31	PARKM	Parking master. 0000 e300 core 0001 PCI, DMA 0010 Reserved 0011 QUICC Engine block 0100 Encryption core 0101–1111Reserved

6.2.2 Arbiter Timers Register (ATR)

The arbiter timers register (ATR) defines the arbiter address time out (ATO) and data time out (DTO) values. Figure 6-2 shows the fields of ATR.

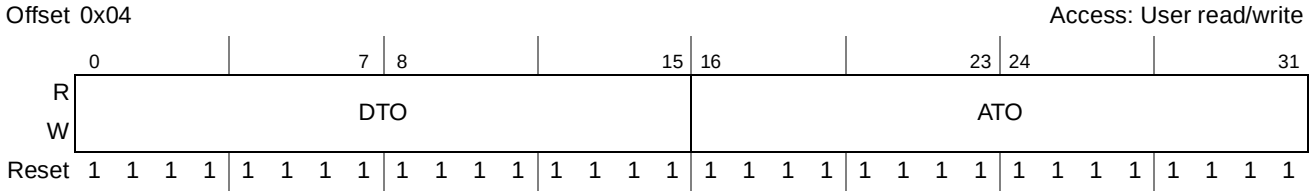


Figure 6-2. Arbiter Timers Register (ATR)

Table 6-3 describes ATR fields.

Table 6-3. ATR Field Descriptions

Bits	Name	Description
0–15	DTO	Data time out. Specifies the time-out period for the data tenure. The granularity of this field is128 bus clocks. The maximum value is8355840 coherent system bus clocks. Data time_out occurs if the data tenure does not end before the specified time-out period. When DTO = n, the timeout cycle is n*128. 0000 Reserved 0001 128 clock cycles 0002 256 clock cycles 0003 384 clock cycles ... FFFF 8355840 clock cycles
16–31	ATO	Address time out. Specifies the time-out period for the address tenure. The granularity of this field is128 bus clocks. Maximum value is8355840 coherent system bus clocks. Address time-out occurs if the address tenure did not end before the specified time-out period. When ATO = n, the timeout cycle is n*128. 0000 Reserved 0001 128 clock cycles 0002 256 clock cycles 0003 384 clock cycles ... FFFF 8355840 clock cycles

6.2.3 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.
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	—		EVENT	—	MSTR_ID	—	TBST	TSIZE	—	TTYPE
W										
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 I2C (boot sequencer) 01010 JTAG 01011 Reserved 01100 Reserved 01101 PCI 01110 Reserved 01111 DMA 10000–11111 QUICC Engine block as follows: 100xy are used by the QUICC Engine block as follows: x = MSTR_ID[3] Least significant bit of SNUM (See Chapter 20, “QUICC Engine Block Control,” 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 11xxx Reserved **Note:** Master Id reflects the source of transaction and is used for debug purpose.
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.

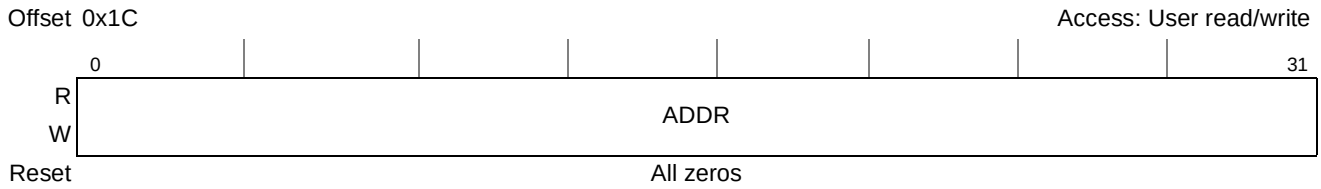


Figure 6-7. Arbiter Event Address Register (AEADR)

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.

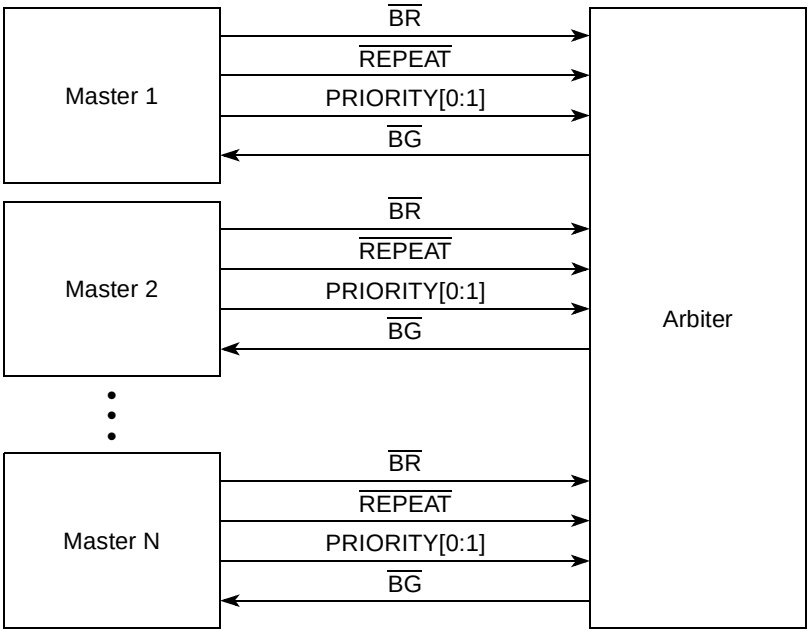


Figure 6-9. Address Bus Arbitration

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

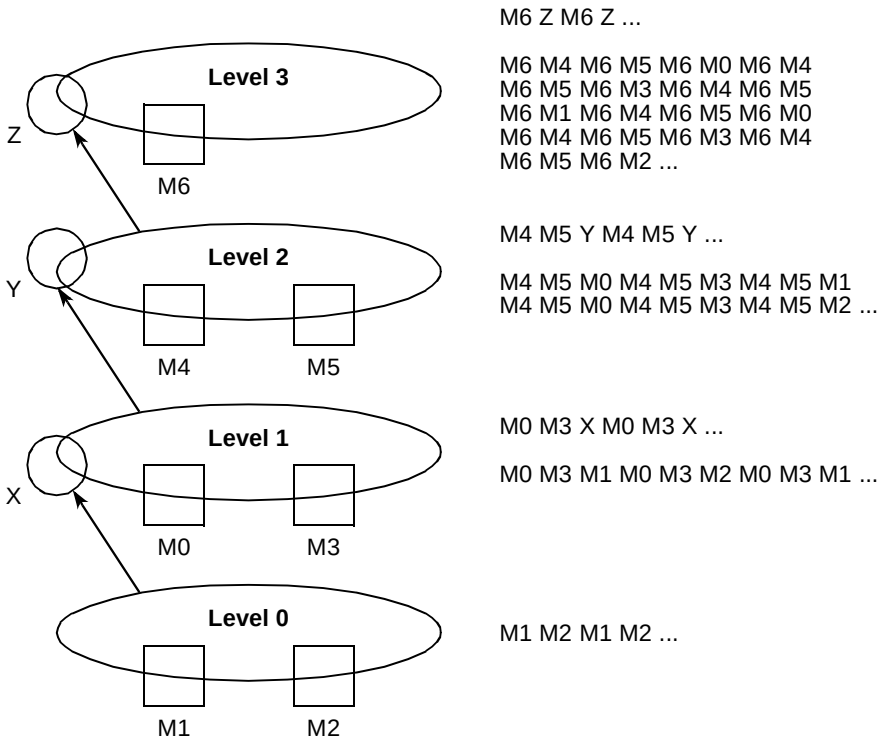


Figure 6-10. An Example of Priority-Based Arbitration Algorithm

NOTE

See each bus master's chapter and Section 5.4.3.4, "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
00001	lwarx reservation set
01001	tlbsync

Table 6-10. Address Only Transaction Type Encoding (continued)

ttype[0:4]	Bus Commands
01101	icbi

The arbiter allows address-only (AO) transactions on the bus and the G2 core has ability to issue address-only (AO) transactions (see HID0 [ABE] in the *G2 PowerPC Core Reference Manual*, Rev 1). As there is no advantage in using AO transaction in this system, the bus monitor allows the detection of AO transactions and treats them as an error.

The transaction with an address only transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[AO].
3. Issues reset request, MCP or regular interrupt according to AERR[AO] and AIDR[AO] if enabled by AMR[AO].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.5 Reserved Transaction Type

Table 6-11 shows transaction types defined as reserved:

Table 6-11. Reserved Transaction Type Encoding

ttype[0:4]	Bus Commands
00101	Reserved
1xx01	Reserved for customer
10110	Reserved
00011	Reserved
00111	Reserved
01111	Reserved
1xx11	Reserved for customer

The transaction with a reserved transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Reports on the event to AER[RES].
3. Issues reset request, MCP or regular interrupt according to AERR[RES] and AIDR[RES], if enabled by AMR[RES].
4. Updates transaction attributes and address of AEATR and AEADR for the first error event.

6.3.2.6 Illegal (eciwx/ecowx) Transaction Type

Table 6-12 shows transaction types defined as illegal.

Table 6-12. Illegal Transaction Type Encoding

ttype[0:4]	Bus command
10100	External control word write (**ecowx**)
11100	External control word read (**eciwx**)

The transaction with an illegal (eciwx, ecowx) transfer type, the arbiter performs as follows:

1. Ends the address tenure by asserting $\overline{\text{AACK}}$.
2. Starts data tenure and ends data tenure by asserting $\overline{\text{TEA}}$.
3. Reports on the event in AER[ECW].
4. Issues reset request, MCP or regular interrupt according to AERR[ECW] and AIDR[ECW], if enabled by AMR[ECW].
5. Updates transaction attributes and address of AEATR and AEADR for the first error event.

See Section 6.2.3, "Arbiter Event Register (AER)," Section 6.2.4, "Arbiter Interrupt Definition Register (AIDR)," Section 6.2.5, "Arbiter Mask Register (AMR)," Section 6.2.6, "Arbiter Event Attributes Register (AEATR)," Section 6.2.7, "Arbiter Event Address Register (AEADR)," and Section 6.2.8, "Arbiter Event Response Register (AERR)," for more information.

6.4 Initialization/Applications Information

The following sections describe the initialization and error handling sequences for the arbiter.

6.4.1 Initialization Sequence

The following initialization sequence is recommended:

1. Write to ACR to configure pipeline depth, address bus parking mode, global maximum repeat count, PCI maximum repeat count.
2. Write to AERR defines whether different error events cause a reset request or an interrupt.
3. Write to AIDR defines the kind of interrupt (regular or MCP) caused by each error event. Note that this is necessary only if interrupts are enabled and AERR defines error events to cause interrupt.
4. Write to AMR to enable interrupts.
5. Write to ATR to set the ATO and DTO timers. Note that this is only necessary if the required timers are less than the maximum value (which is default).

6.4.2 Error Handling Sequence

The following error handling sequence is recommended:

1. Read to AER to find out about the error that occurred in the system. Also, read the values of AEATR and AEADR to check on the first error event in the system.
2. If those registers are not accessible because of a bus deadlock situation, reset the chip and read the values of the AEATR and AEADR registers to check on the event that causes this problem to the system. Use $\overline{\text{HRESET}}$ to reset the chip to guarantee that the information stored in AEATR and AEADR is not lost.
3. Clear all the previous events by writing ones to the AER. This register is also cleared after reset.

Chapter 7
e300 Processor Core Overview

This chapter provides an overview of features for the embedded microprocessors in the e300 core family, which are PowerPC™ microprocessors built on Power Architecture™ technology. Throughout this document, the terms ‘e300 core’, ‘core’, and ‘processor’ are used interchangeably. The term ‘e300c1’ is used when describing an implementation-specific feature or when a difference exists between different configurations. The term ‘e300’ is used when describing a feature that pertains to the family of e300 processors. The MPC8360 uses an e300c1 core.

7.1 Overview

This section describes the details of the e300 core, provides a block diagram showing the major functional units, and briefly describes how these units interact. All differences between the e300 and previous PowerPC implementations derived from the MPC603e processor are noted. For additional information, please refer to the *e300 PowerPC Core Family Reference Manual*.

The e300 core is a low-power implementation of this microprocessor family of reduced instruction set computing (RISC) microprocessors. The core implements the 32-bit portion of the PowerPC architecture, which defines 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The core is a superscalar processor that can issue and retire as many as three instructions per clock cycle. Instructions can execute out of program order for increased performance; however, the core makes completion appear sequential.

The e300 core integrates independent execution units including: an integer unit (IU) a floating-point unit (FPU), a branch processing unit (BPU), a load/store unit (LSU), and a system register unit (SRU). s The ability to execute instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for e300-core-based systems. Most integer instructions execute in one clock cycle. In thee300c1 core, the FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle. Thee300c1 core provides hardware support for all single- and double-precision floating-point operations for most value representations and all rounding modes.

Figure 7-1 shows a block diagram of the e300c1 core.

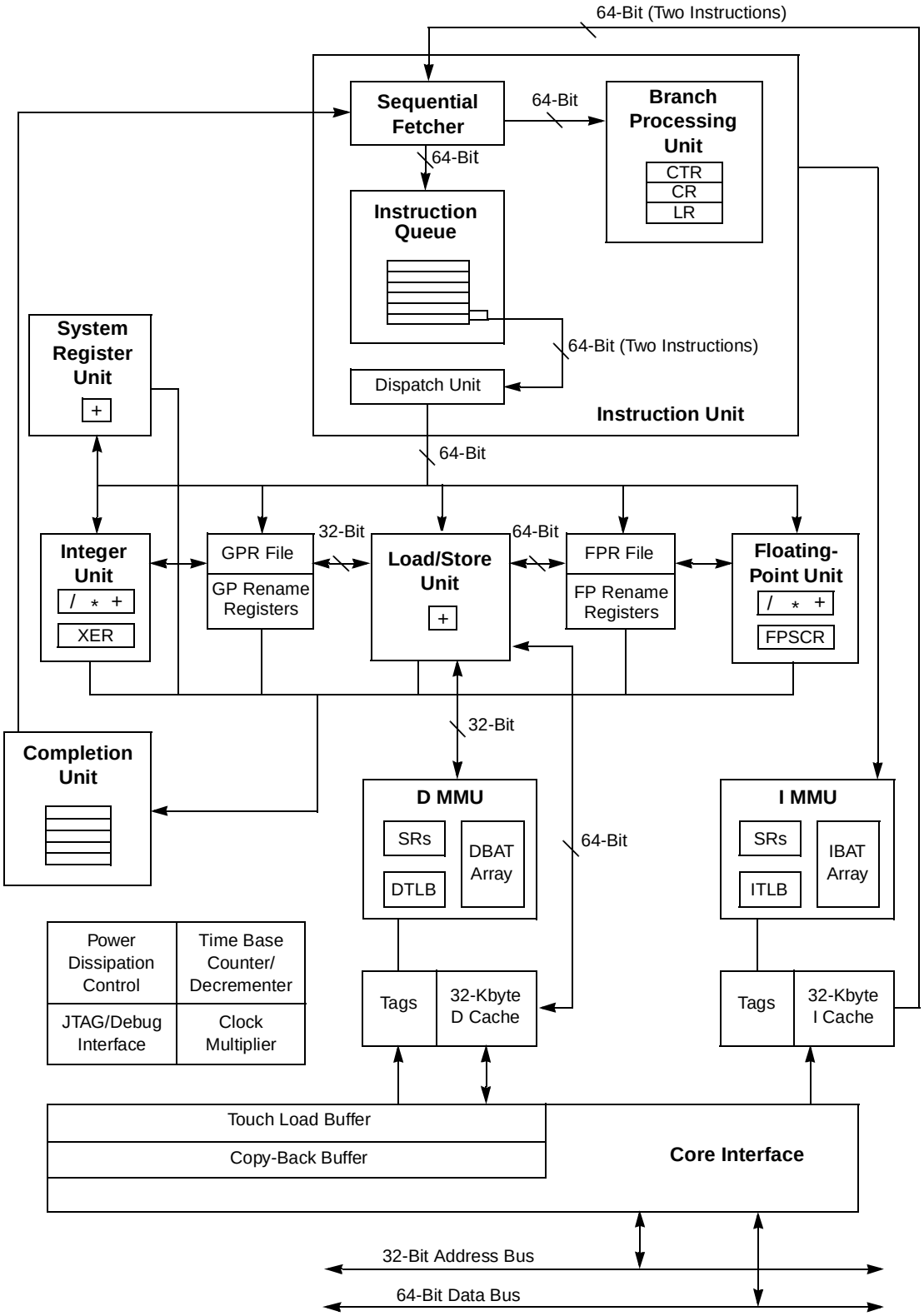


Figure 7-1. e300c1 Core Block Diagram

The e300c1 provides independent, on-chip, 32-Kbyte, eight-way, set-associative, physically-addressed caches for instructions and data, and on-chip instruction and data memory management units (MMUs).s The MMUs contain 64-entry, two-way, set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged, virtual-memory, address translation and variable-sized block translation. The TLBs use a least recently used (LRU) replacement algorithm and the caches use a pseudo least recently used algorithm (PLRU).

The core also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays, each containing eight pairs of BATs, an increase from four pairs of each type of BATs in the G2 core. This increase provides more flexibility in protecting accesses and providing translation on a segment, block, or page basis for memory accesses and I/O accesses. Effective addresses are compared simultaneously with all eight entries in the BAT array during block translation. In accordance with the PowerPC architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

As part of the coherent system bus (CSB), the e300 core has a 64-bit data bus and a 32-bit address bus. During normal operation, the e300 core provides a three-state (modified, exclusive, and invalid) coherency protocol which is a compatible subset of a four-state (modified/exclusive/shared/invalid) MESI protocol. However, the e300 data cache contains a programmable MESI extension that supports the shared cache coherency state (similar to other PowerPC processors). Both protocols operate coherently in systems that contain four-state caches. Although MESI is supported by the e300 core, it is not implemented on the MPC8360. The core also supports single-beat and burst data transfers for memory accesses and supports memory-mapped I/O operations.

The true little-endian mode is another enhanced capability of the e300 core. Unlike the PowerPC little-endian mode (which manipulates only the address bits), no longer supported on the e300, the true little-endian mode actually operates on true little-endian instructions and data from memory.

The critical interrupt is an additional interrupt in the e300 core and has higher priority order than the system management interrupt. Also, debug features are improved in the e300. Additional SPRG interrupt handling registers are provided for enhancing flexibility for the operating system.

7.1.1 Features

This section describes the major features of the e300 core:

- High-performance, superscalar microprocessor core
  - — As many as three instructions issued and retired per clock (two instructions plus one branch instruction)
  - — As many as five instructions in execution per clock
  - — Single-cycle execution for most instructions
  - — Pipelined floating-point unit (FPU) for all single- and double-precision operations
- Independent execution units and two register files
  - — Branch processing unit (BPU) featuring static branch prediction
  - — One 32-bit integer units (IU) in the e300c1.

  - FPU based on the IEEE 754™ standard for both single- and double-precision operations
  - Load/store unit (LSU) for data transfer between data-cache and general-purpose registers (GPRs) and floating-point registers (FPRs)
  - System register unit (SRU) that executes condition register (CR), special-purpose register (SPR), and integer add/compare instructions. Add/compare instructions are also executed in the IUs.
  - Thirty-two 32-bit GPRs for integer operands
  - Thirty-two 64-bit FPRs for single- or double-precision operands
- High instruction and data throughput
  - Zero-cycle branch capability (branch folding)
  - Programmable static branch prediction on unresolved conditional branches
  - Instruction fetch unit capable of fetching two instructions per clock from the instruction cache
  - A six-entry instruction queue (IQ) that provides lookahead capability
  - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
  - 32-Kbyte data cache and 32-Kbyte instruction cache with parity—eight-way, set-associative, physically addressed, PLRU replacement algorithm on thee300c1 .
  - 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 *MPC8360 Integrated Processor Hardware Specifications*
- Integrated power management
  - Internal processor/bus clock multiplier ratios
  - Three power-saving modes: doze, nap, and sleep
  - Automatic dynamic power reduction when internal functional units are idle

- In-system testability and debugging features through JTAG boundary-scan capability

Features specific to the e300 core not present on the G2 processors follow:

- Enhancements to the register set
  - The e300 core has one more HID0 bit than the G2:
    - The enable cache parity checking (ECPE) bit, HID0[1], gives the e300 core the ability to enable the taking of a machine check interrupt based on the detection of a cache parity error
- Enhancements to cache implementation
  - 32-Kbyte, eight-way, set-associative instruction and data cache on thee300c1
  - Full parity checking is performed on both instruction and data cache memory arrays
  - Lockable L1 instruction and data caches—entire cache or on a per-way basis up to 7 of 8 ways on thee300c1
  - New **icbt** instruction supports initialization of instruction cache
  - Data cache supports four-state MESI coherency protocol (not implemented on MPC8360E)
  - The instruction cache is blocked only until the critical load completes (hit under reloads allowed)
  - Instruction cancel mechanism improves utilization of instruction cache by supporting hits-under-cancels and misses-under-cancels.
  - The critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.
  - Data cache queue sharing makes cast-outs and snoop pushes more efficient
  - Provides for an optional data cache operation broadcast feature (enabled by HID0[ABE]) that allows for coherent system management. All of the data cache control instructions, except **dcbz** (**dcbi**, **dcbf**, and **dcbst**) require that HID0[ABE] be enabled to broadcast.
  - Instruction fetch burst feature allows all instruction fetches from caching-inhibited space to be performed on the bus as burst transactions
- Interrupts
  - The e300 core offers hardware support for misaligned little-endian accesses. Little-endian load/store accesses that are not on a word boundary, except for strings and multiples, generate interrupts under the same circumstances as big-endian accesses.
  - The e300 core supports true little-endian mode to minimize the impact on software porting from true little-endian systems.
  - An input interrupt signal, $\overline{cint}$, is provided to trigger the critical interrupt exception on the e300 core.
- Bus clock—PLL configuration signals include seven signals for settings and control: *pll_cfg[0:6]*.
- Debug features
  - Breakpoint status recorded in DBCR and IBCR control registers
  - Two signals for the debug interface: $\overline{stopped}$ and $\overline{ext\_halt}$

Figure 7-1, provides a block diagram of the e300 core that shows how the execution units—IU, FPU, BPU, LSU, and SRU—operate independently and in parallel. It should be noted that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the device.

The e300 core provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing are handled in the instruction unit. Translation of addresses for cache or external memory accesses are handled by the MMUs. Both units are discussed in more detail in Section 7.1.2, "Instruction Unit," and Section 7.1.5.1, "Memory Management Units (MMUs)."

7.1.2 Instruction Unit

As shown in Figure 7-1, , the e300 core instruction unit, containing a fetch unit, instruction queue, dispatch unit, and BPU, provides centralized control of instruction flow to the execution units. The instruction unit determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The instruction unit fetches the instructions from the instruction cache into the instruction queue. The BPU receives branch instructions from the fetcher and uses static branch prediction to allow fetching from a predicted instruction stream while a conditional branch is evaluated. The BPU folds out for unconditional branch instructions and conditional branch instructions unaffected by instructions in the execution pipeline.

Instructions issued beyond a predicted branch cannot complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these are branch instructions, they are decoded but not issued. Instructions to be executed by the FPU, IU, LSU, and SRU are issued and allowed to progress up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and execution continues along the predicted path.

If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

7.1.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 7-1, , holds as many as six instructions and loads up to two instructions from the instruction unit during a single cycle. The instruction fetch unit continuously loads as many instructions as space in the IQ allows. Instructions are dispatched to their respective execution units from the dispatch unit at a maximum rate of two instructions per cycle. Dispatching is facilitated to the IUs, FPU, LSU, and SRU by the provision of a reservation station at each unit. The dispatch unit performs source and destination register dependency checking, determines dispatch serializations, and inhibits subsequent instruction dispatching as required.

For a more detailed overview of instruction dispatch, see Section 7.3.6, "Instruction Timing."

7.1.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the fetch unit and performs CR lookahead operations on conditional branches to resolve them early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the core fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder to compute branch target addresses and three user-control registers: the link register (LR), the count register (CTR), and the conditional register (CR). The BPU calculates the return pointer for sub-routine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**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. s

7.1.3.2 Floating-Point Unit (FPU)

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the core to efficiently implement multiply and multiply-add operations. The FPU is pipelined so that single- and double-precision instructions can be issued back-to-back. The 32 FPRs are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by the automatic allocation of rename registers. The core writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

Thee300c1 core supports all floating-point data types based on the IEEE 754 standard (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software interrupt routines.

7.1.3.3 Load/Store Unit (LSU)

The LSU executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

Load and store instructions are issued and executed in program order; however, the memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering.

Cacheable loads, when free of data bus dependencies, can execute out of order with a maximum throughput of one per cycle and with a two-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR or FPR. Stores cannot be executed in a predicted manner and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The core executes store instructions with a maximum throughput of one per cycle and with a three-cycle total latency. The time required to perform the actual load or store depends on whether the operation involves the cache, system memory, or an I/O device.

7.1.3.4 System Register Unit (SRU)

The SRU executes various system-level instructions, including condition register logical operations and move to/from special-purpose register instructions. It also executes integer add/compare instructions. In order to maintain system state, most instructions executed by the SRU are completion-serialized; that is, the instruction is held for execution in the SRU until all prior instructions issued have completed. Results from completion-serialized instructions executed by the SRU are not available or forwarded for subsequent instructions until they complete.

7.1.4 Completion Unit

The completion unit tracks instructions in program order from dispatch through execution and then completes. Completing an instruction commits the core to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the core must recover from a mispredicted branch or an interrupt.

Instruction state and other information required for completion is kept in a five-entry FIFO completion queue. A single completion queue entry is allocated for each instruction once it enters the execution unit from the dispatch unit. An available completion queue entry is a required resource for dispatch; if no completion entry is available, dispatch stalls. A maximum of two instructions per cycle are completed in order from the queue.

7.1.5 Memory Subsystem Support

The core provides separate instruction and data caches and MMUs. The core also provides an efficient processor bus interface to facilitate access to main memory and other bus subsystems. The memory subsystem support functions are described in the following sections.

7.1.5.1 Memory Management Units (MMUs)

The core MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory (referred to as real memory in the architecture specification) for instruction and data. The MMUs also control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to assist implementation of a demand-paged virtual memory system. Note that software assistance is required for the device to maintain reference and changed status. A key bit is implemented to provide information about memory protection violations prior to page table search operations.

The LSU calculates effective addresses for data loads and stores, performs data alignment to and from cache memory, and provides the sequencing for load and store string and multiple word instructions. The instruction unit calculates effective addresses for instruction fetching.

After an EA is generated, its higher-order bits are translated by the appropriate MMU into physical address bits. The lower-order EA bits are the same on the physical address which are directed to the on-chip cache and formed the index into a four-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is then used by the memory unit and the core interface to access external memory.

The MMU also directs the address translation and enforces the protection hierarchy programmed by the operating system in relation to the supervisor/user privilege level of the access and in relation to whether the access is a load or store.

For instruction fetches, the IMMU looks for the address in the ITLB and in the IBAT array. If an address hits both, the IBAT array translation is used. Data accesses cause a lookup in the DTLB and DBAT array. In most cases, the translation is in a TLB and the physical address bits are available to the on-chip cache.

The e300 core implements four more IBAT and four more DBAT entries than the G2.

When the EA misses in the TLBs, the core provides hardware assistance for software to perform a search of the translation tables in memory. The hardware assist consists of the following features:

- Automatic storage of the missed effective address in IMISS and DMISS
- Automatic generation of the primary and secondary hashed real addresses of the page-table entry group (PTEG), which are readable from the HASH1 and HASH2 register locations.

  The HASH data is generated from the contents of the IMISS or DMISS register. The register that is selected depends on the miss (instruction or data) that was last acknowledged.
- Automatic generation of the first word of the page table entry (PTE) of the tables being searched
- A real page address (RPA) register that matches the format of the lower word of the PTE
- TLB access instructions (**tlbli** and **tlbld**) that are used to load an address translation into the instruction or data TLBs
- Shadow registers for GPR0–GPR3 that allow miss code to execute without corrupting the state of any of the existing GPRs. Shadow registers are used only for servicing a TLB miss.

See Section 7.3.5.2, “Implementation-Specific Memory Management,” for more information about memory management for the core.

7.1.5.2 Cache Units

The e300c1 provides independent, 32-Kbyte, eight-way, set-associative, instruction and data caches. s 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 *qreq* signal and will only enter another power mode after an acknowledge (*qack*) is received. The four power modes are as follows:

- Full-power—This is the default power state of the e300 core. The e300 core is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware.
- Doze—All the functional units of the e300 core are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, system management interrupt, 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 (*mcp*) signal. A return to full-power state from a nap state takes only a few processor clock cycles.
- Sleep—Sleep mode reduces power consumption to a minimum by disabling all internal functional units; then external system logic may disable the PLL and *sysclk*. Returning the core to the full-power state requires the enabling of the PLL and *sysclk,* followed by the assertion of an external asynchronous interrupt, system management interrupt, hard or soft reset, or *mcp* signal after the time required to relock the PLL.

7.1.7.2 Time Base/Decrementer

The time base is a 64-bit register (accessed as two 32-bit registers) that is incremented once every four bus clock cycles; external control of the time base is provided through the time base 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.”

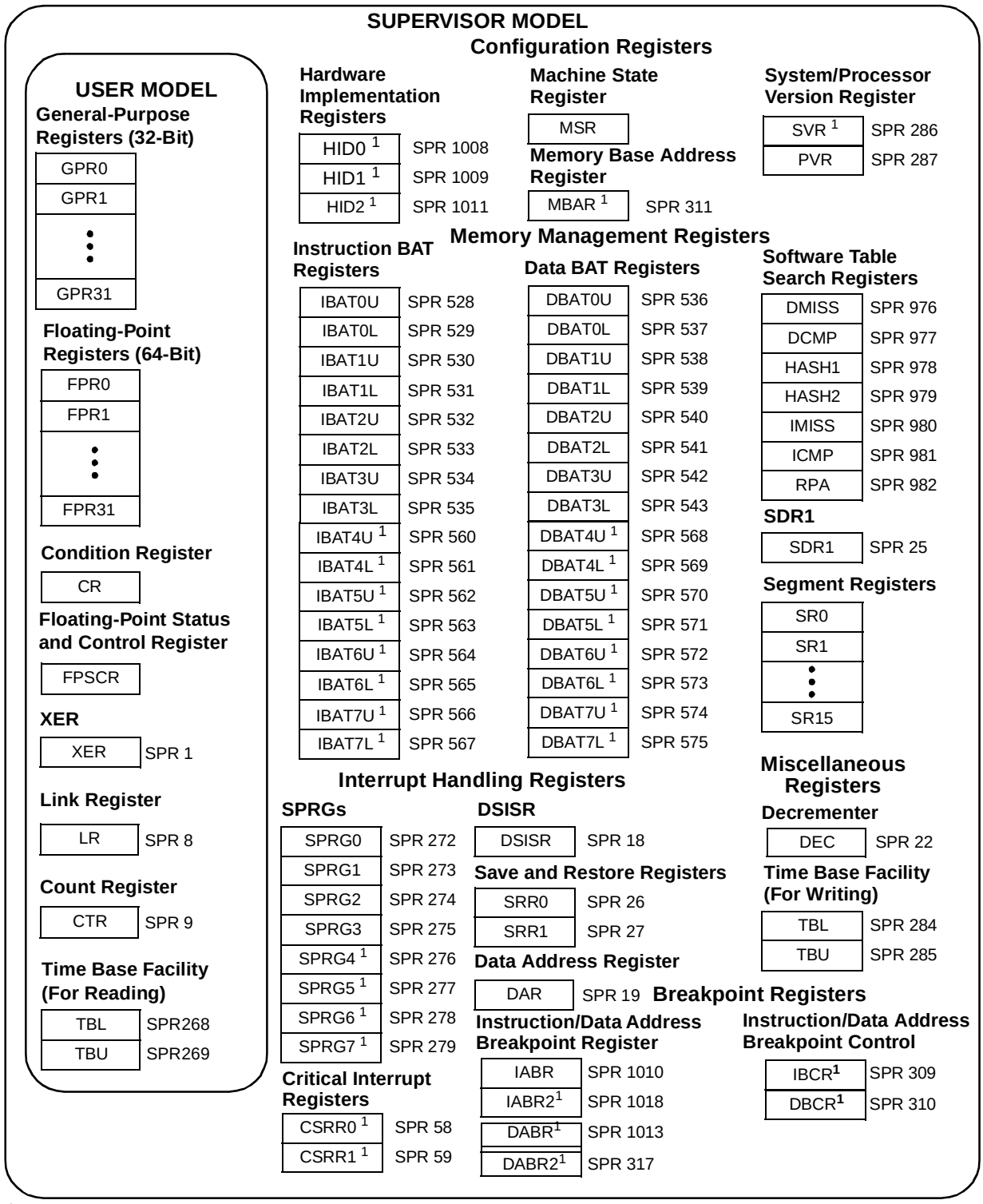
7.3.1 Register Model

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.

7.3.1.1.3 Condition Register (CR)

The CR is a 32-bit user-level register that provides a mechanism for testing and branching. It consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point comparisons, arithmetic, and logical operations.

7.3.1.1.4 Floating-Point Status and Control Register (FPSCR)

The user-level FPSCR contains all floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE 754 standard.

7.3.1.1.5 User-Level SPRs

The PowerPC architecture defines numerous special purpose registers that serve a variety of functions, such as providing controls, indicating status, configuring the core, and performing special operations. During normal execution, a program can access the registers, as shown in Figure 7-2, depending on the program's access privilege (supervisor or user, determined by the privilege-level bit, MSR[PR]). Note that GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**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.
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

Table 7-1. MSR Bit Descriptions (continued)

Bits	Name	Description
18	FP	Floating-point available 0 The processor prevents dispatch of floating-point instructions, including floating-point loads, stores, and moves. 1 The processor can execute floating-point instructions and can take floating-point enabled exception type program interrupts.
19	ME	Machine check enable 0 Machine check interrupts are disabled 1 Machine check interrupts are enabled
20	FE0	Floating-point exception mode 0
21	SE	Single-step trace enable 0 The processor executes instructions normally 1 The processor generates a trace interrupt upon the successful completion of the next instruction
22	BE	Branch trace enable 0 The processor executes branch instructions normally 1 The processor generates a trace interrupt upon the successful completion of a branch instruction
23	FE1	Floating-point exception mode 1
24	CE	Critical interrupt enable 0 Critical interrupts disabled 1 Critical interrupts enabled; critical interrupt and **rfci** instruction enabled The critical interrupt is an asynchronous implementation-specific interrupt. The critical interrupt vector offset is 0x00A00. The **rfci** instruction is implemented to return from these interrupt handlers. Also, CSRR0 and CSRR1 are used to save and restore the processor state for critical interrupts.
25	IP	Interrupt prefix. The setting of this bit specifies whether an interrupt vector offset is prepended with Fs or 0s. In the following description, *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.
30	RI	Recoverable interrupt (for system reset and machine check interrupts) 0 Interrupt is not recoverable 1 Interrupt is recoverable
31	LE	Little-endian mode enable 0 The processor runs in big-endian mode 1 The processor runs in little-endian mode.

[1] All reserved bits should be set to zero for future compatibility.

7.3.1.3.2 Segment Registers (SRs)

For memory management, 32-bit processors implement sixteen 32-bit SRs. To speed access, the core implements the SRs as two arrays: a main array, for data memory accesses, and a shadow array, for

instruction memory accesses. Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays.

7.3.1.3.3 Supervisor-Level SPRs

The e300 core, like the G2_LE core, has additional supervisor-level SPRs, which are shown in Figure 7-2. Two critical interrupt SPRs (CSRR0 and CSRR1), eight SPRGs (SPRG0–SPRG7), eight pairs of instruction BATs (IBAT0–IBAT7), eight pairs of data BATs (DBAT0–DBAT7), one system version register (SVR), one system memory base address (MBAR), one instruction address breakpoint control (IBCR), one data address breakpoint control (DBCR), a new instruction breakpoint register (IABR2), and two data address breakpoint registers (DABR and DABR2) are integrated into the core.

Supervisor-level SPRs include the following:

- The DSISR defines the cause of data access and alignment interrupts. The cause of a DSI interrupt for a data breakpoint (match with DABR and DABR2) can be determined by the value of the DSISR[DABR] bit (bit 9).
- The data address register (DAR) holds the address of an access after an alignment or DSI interrupt. For example, it contains the address of the breakpoint match condition.
- The 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 $\overline{mcp}$. The purpose of this bit is to mask out machine check interrupts caused by assertion of $\overline{mcp}$, similar to how MSR[EE] can mask external interrupts. 0 Masks $\overline{mcp}$. Asserting $\overline{mcp}$ does not generate a machine check interrupt or a checkstop. 1 Asserting $\overline{mcp}$ causes checkstop if MSR[ME] = 0 or a machine check interrupt if ME = 1
1	ECPE	Enable cache parity errors. 0 Disables instruction and data cache parity error reporting 1 Allows a detected cache parity error to cause a machine check interrupt if MSR[ME] = 1 or a checkstop if MSR[ME] = 0
2	EBA	Enable *ap_in[0:3]* and $\overline{ape}$ for address parity checking. 0 Disables address parity checking during a snoop operation 1 Allows an address parity error during snoop operations to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1
3	EBD	Enable $\overline{dpe}$ for data parity checking. 0 Disables data parity checking 1 Allows a data parity error during reads to cause a checkstop if MSR[ME] = 0 or a machine check interrupt if MSR[ME] = 1

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
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*.
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, *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, *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

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
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.
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

Table 7-2. e300 HID0 Bit Descriptions (continued)

Bits	Name	Function
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 HID1[ECLK] and HID1[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.
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	
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.

Table 7-5. e300HID2 Bit Descriptions (continued)

Bits	Name	Description
16–18	IWLCK[0–2]	Instruction cache way-lock. Useful for locking blocks of instructions into the instruction cache for time-critical applications that require deterministic behavior. 000 = no ways locked 001 = way 0 locked 010 = way 0 through way 1 locked 011 = way 0 through way 2 locked 100 = way 0 through way 3 locked in e300c1 . 101 = way 0 through way 4 locked in e300c1 . 110 = way 0 through way 5 locked in e300c1 . 111 = way 0 through way 6 locked in e300c1 . Setting HID0[ILOCK] will lock all ways.
19	ICWP	Instruction cache way protection. Used to protect locked ways in the instruction cache from being invalidated. 0 Instruction cache way protection disabled 1 Instruction cache way protection enabled
20–23	—	Reserved, should be cleared.
24–26	DWLCK[0–2]	Data cache way-lock. Useful for locking blocks of data into the data cache for time-critical applications where deterministic behavior is required. 000 = no ways locked 001 = way 0 locked 010 = way 0 through way 1 locked 011 = way 0 through way 2 locked 100 = way 0 through way 3 locked in e300c1 . 101 = way 0 through way 4 locked in e300c1 . 110 = way 0 through way 5 locked in e300c1 . 111 = way 0 through way 6 locked in e300c1 . Setting HID0[DLOCK] will lock all ways.
27–31	—	Reserved, should be cleared.

7.3.2 Instruction Set and Addressing Modes

The following sections describe the PowerPC instruction set and addressing modes in general.

7.3.2.1 PowerPC Instruction Set and Addressing Modes

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
  - — Integer arithmetic instructions
  - — Integer compare instructions
  - — Integer logical instructions
  - — Integer rotate and shift instructions

- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load/store instructions—These include integer and floating-point load and store instructions.
  - Integer load and store instructions
  - Integer load and store multiple instructions
  - Floating-point load and store
  - Primitives used to construct atomic memory operations (**lwarx** and **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 e300c1 provide independent, 32-Kbyte, eight-way, set-associative, instruction and data caches.provides 16-Kbyte, four-way set-associative instruction and data caches. The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The data cache is configured as 128 sets of 8 blocks each on the e300c1 . Each block consists of 32 bytes, 2 state bits, and an address tag. The two state bits implement the three-state MEI (modified/exclusive/invalid) protocol. Each block contains eight 32-bit words. Note that the PowerPC architecture defines the term ‘block’ as the cacheable unit. For the core, the block size is equivalent to a cache line. A block diagram of the data cache organization is shown in Figure .

The instruction cache is configured as 128 sets of 8 blocks each on the e300c1 . Each block consists of 32 bytes, an address tag, and a valid bit. The instruction cache may not be written to, except through a block fill operation. In the e300 core, the instruction cache is blocked only until the critical load completes. The e300 core supports instruction fetching from other instruction cache lines following the forwarding of the critical-first-double-word of a cache line load operation. Successive instruction fetches from the cache line being loaded are forwarded, and accesses to other instruction cache lines can proceed during the cache line load operation. The instruction cache is not snooped, and cache coherency must be maintained by software. A fast hardware invalidation capability is provided to support cache maintenance. The organization of the instruction cache for the e300c1 is very similar to the data cache shown in Figure 7-3.

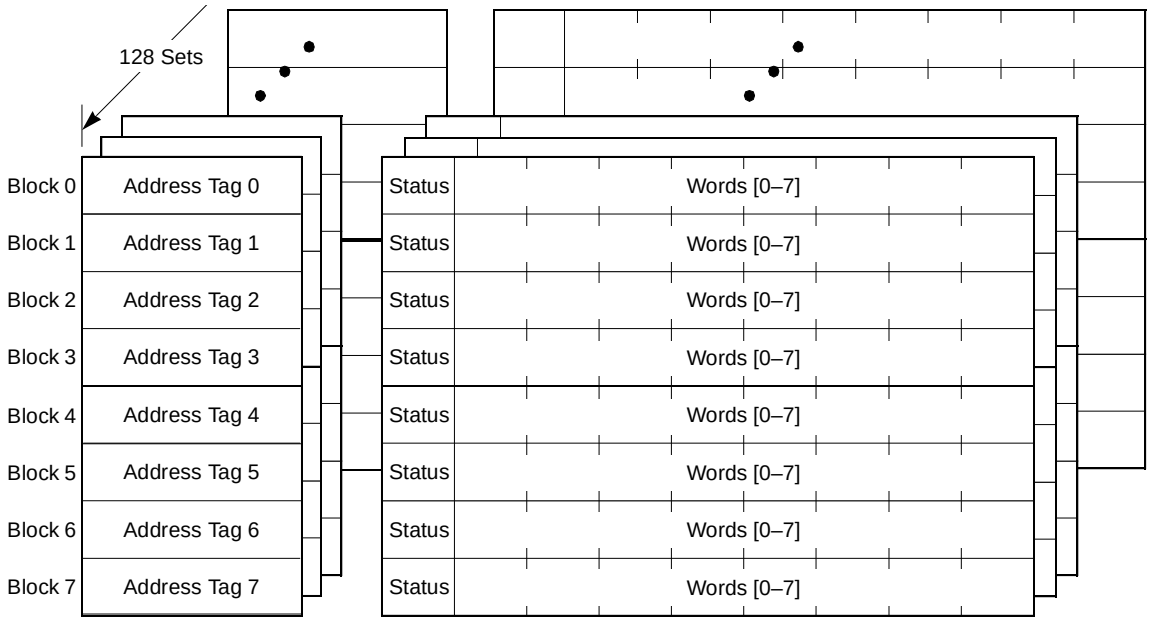


Figure 7-3. e300c1 Data Cache Organization

Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A[27–31] of the effective addresses are zero); thus, a cache block never crosses a page boundary. Misaligned accesses across a page boundary can incur a performance penalty.

The e300 core cache blocks are loaded in four beats of 64 bits each on the 64-bit data bus. The burst load is performed as critical-double-word-first. The data cache is blocked to internal accesses until the load

completes; the instruction cache allows sequential fetching during a cache block load. In the core, the critical-double-word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the core implements the MEI protocol during normal operation of the data cache. The new data cache MESI extension supports the additional fourth cache coherency shared state for the data cache. To support this feature, the shared signal, *shd*, has been added to the bus interface. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8360. The following four states indicate the state of the cache block:

- Modified—The cache block is modified with respect to system memory; that is, data for this address is valid only in the cache and not in system memory.
- Exclusive—This cache block holds valid data that is identical to the data at this address in system memory. No other cache has this data.
- Shared—Only available if HID2[MESISTATE] register bit is set. The address block is valid in the cache and in at least one other cache. This block is always consistent with system memory. That is, the shared state is shared-unmodified; there is no shared-modified state. Although the MESI protocol is supported by the e300 core, it is not implemented on MPC8360.
- Invalid—This cache block does not hold valid data.

Cache coherency is enforced by on-chip bus snooping logic. Because the e300 core data cache tags are single-ported, a simultaneous load/store and snoop access represents a resource contention. The snoop access is given first access to the tags. The load or store then occurs on the clock following the snoop.

Parity is now integrated into both instruction and data cache memory. A machine check interrupt is now taken upon the detection of an instruction or data cache parity error. Parity is checked whenever valid data is returned from the instruction or data cache for a cache hit or whenever valid data is read out of the cache for a castout or snoop-push operation.

7.3.3.3 Instruction and Data Cache Way-Locking

The e300 core implements instruction and data cache way-locking, which guarantees that certain memory accesses will hit in the cache. This provides deterministic access times for those accesses.

7.3.4 Interrupt Model

This section describes the PowerPC interrupt model and the e300 core implementation specifically.

7.3.4.1 PowerPC Interrupt Model

The PowerPC interrupt mechanism allows the core to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. The conditions that can cause interrupts are called exceptions. When interrupts occur, information about the state of the core is saved to certain registers and the core begins execution at an address (interrupt vector) predetermined for each interrupt type. Interrupts are processed in supervisor mode.

Some interrupts, such as program interrupts, can be triggered by a broad range of exception conditions. Other interrupts, such as the 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 $\overline{sreset}$ or $\overline{hreset}$.
Machine check	00200	Caused by the assertion of the $\overline{tea}$ signal during a data bus transaction, assertion of $\overline{mcp}$, an address or data parity error, or an instruction or data cache parity error. Note that the e300 has SRR1 register values that are different from the G2/G2_LE cores' when a machine check occurs.
DSI	00300	Determined by the bit settings in the DSISR, listed as follows: 1 Set if the translation of an attempted access is not found in the primary hash table entry group (HTEG), or in the rehashed secondary HTEG, or in the range of a DBAT register; otherwise cleared 4 Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared 6 Set for a store operation and cleared for a load operation 9 Set if a data address breakpoint interrupt occurs when the data [0–28] in the DABR or DABR2 matches the next data access (load or store instruction) to complete in the completion unit. The different breakpoints are enabled as follows: • Write breakpoints enabled when DABR[30] is set • Read breakpoints enabled when DABR[31] is set
ISI	00400	Caused when an instruction fetch cannot be performed for any of the following reasons: • The effective (logical) address cannot be translated. That is, there is a page fault for this portion of the translation, so an ISI interrupt must be taken to load the PTE (and possibly the page) into memory. • The fetch access violates memory protection (indicated by SRR1[4] set). If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.
External interrupt	00500	Caused when MSR[EE] = 1 and the $\overline{int}$ signal is asserted.
Alignment	00600	Caused when the core cannot perform a memory access for any of the reasons described below: • The operand of a floating-point load or store instruction is not word-aligned. • The operands of **lmw**, **stmw**, **lwarx**, and **stwcx.** instructions are not aligned. • The instruction is **lswi**, **lswx**, **stswi**, **stswx**, and the core is in little-endian mode. Note that PowerPC little-endian mode is not supported on the e300 core. • The operand of **dcbz** is in memory that is write-through-required or caching-inhibited.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
Program	00700	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.
Decrementer	00900	Occurs when DEC[31] changes from 0 to 1. This interrupt is enabled with MSR[EE].
Critical interrupt	00A00	Taken when $\overline{cint}$ is asserted and MSR[CE] = 1.
Reserved	00B00–00BFF	—
System call	00C00	Occurs when a System Call (**sc**) instruction is executed.
Trace	00D00	Taken when MSR[SE] =1 or when the currently completing instruction is a branch and MSR[BE] =1.
Reserved	00E00	The e300 core does not generate an interrupt to this vector. Other devices may use this vector for floating-point assist interrupts.
Instruction translation miss	01000	Caused when the effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	Caused when the effective address for a data load operation cannot be translated by the DTLB.
Data store translation miss	01200	Caused when the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs and the change bit in the PTE must be set due to a data store operation.
Instruction address breakpoint	01300	Occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit, and IABR[30] is set. Note that the e300 core also implements IABR2, which functions identically to IABR.

Table 7-7. Exceptions and Interrupts (continued)

Interrupt Type	Vector Offset (hex)	Exception Conditions
System management interrupt	01400	Caused when MSR[EE] = 1 and the $\overline{smi}$ input signal is asserted.
Reserved	01500–02FFF	—

7.3.5 Memory Management

The following sections describe the memory management features of the PowerPC architecture and the e300 core implementation, respectively.

7.3.5.1 PowerPC Memory Management

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses and to provide access protection on blocks and pages of memory.

The core generates two types of accesses that require address translation: instruction accesses and data accesses to memory generated by load and store instructions.

The PowerPC MMU and interrupt model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages are loaded into physical memory from system memory only when they are first accessed by an executing program.

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

The page table contains a number of page-table entry groups (PTEGs). A PTEG contains eight page-table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Address translations are enabled by setting bits in the MSR—MSR[IR] enables instruction address translations, and MSR[DR] enables data address translations.

7.3.5.2 Implementation-Specific Memory Management

The instruction and data memory management units in the e300 core provide 4 Gbytes of logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. Block sizes range from 128 Kbytes to 256 Mbytes and are software selectable. In addition, the core uses an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses. The MMUs in the e300 core rely on the interrupt processing mechanism for the implementation of the paged virtual memory environment and for enforcing protection of designated memory areas.

Instruction and data TLBs provide address translation in parallel with the on-chip cache access, incurring no additional time penalty in the event of a TLB hit. A TLB is a cache of the most recently used page table

entries. Software is responsible for maintaining the consistency of the TLB with memory. The core TLBs are 64-entry, two-way, set-associative caches that contain instruction and data address translations. The core provides hardware assist for software table search operations through the hashed page table on TLB misses. Supervisor software can invalidate TLB entries selectively.

For instructions and data that correspond to block address translation, the e300 core provides independent eight-entry BAT arrays. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software. HID2[HBE] is added to the e300 for enabling or disabling the four additional pairs of BAT registers. However, regardless of the setting of HID2[HBE], these BATs are accessible by **mfspr** and **mtspr**.

As specified by the PowerPC architecture, the hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

Also as specified by the PowerPC architecture, the page table contains a number of PTEGs. A PTEG contains 8 PTEs of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

7.3.6 Instruction Timing

The e300 core is a pipelined superscalar processor core. Because instruction processing is reduced into a series of stages, an instruction does not require all of the resources of an execution unit at the same time. For example, after an instruction completes the decode stage, it can pass on to the next stage, while the subsequent instruction can advance into the decode stage. This improves the throughput of the instruction flow. For example, it may take three cycles for a single floating-point instruction to execute, but if there are no stalls in the floating-point pipeline, a series of floating-point instructions can have a throughput of one instruction per cycle.

The core instruction pipeline has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. Additionally, if possible, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage.
- The dispatch pipeline stage is responsible for decoding the instructions supplied by the instruction fetch stage and determining which of the instructions are eligible to be dispatched in the current cycle. In addition, the source operands of the instructions are read from the appropriate register file and dispatched with the instruction to the execute pipeline stage. At the end of the dispatch pipeline stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- In the execute pipeline stage, each execution unit with an instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage when the execution has finished. In the case of an internal interrupt, the execution unit reports the interrupt to the completion/write-back pipeline stage and discontinues instruction execution until the interrupt is handled. The interrupt is not signaled until that instruction is the next to be completed. Execution of most floating-point instructions is pipelined within the FPU, allowing up to three instructions to execute in the FPU concurrently. The FPU pipeline stages are multiply, add, and round-convert. The LSU has two

pipeline stages: the first stage, for effective address calculation and MMU translation, and the second, for accessing data in the cache.

- The complete/write-back pipeline stage maintains the correct architectural machine state and transfers the contents of the rename registers to the GPRs and FPRs as instructions are retired. If the completion logic detects an instruction causing an interrupt, all subsequent instructions are canceled, their execution results in rename registers are discarded, and instructions are fetched from the correct instruction stream.

A superscalar processor core issues multiple, independent instructions into multiple pipelines, allowing instructions to execute in parallel. The e300c1 core has independent execution units for: integer instructions, floating-point instructions, branch instructions, load/store instructions, and system register instructions.ss The IU and the FPU each have dedicated register files for maintaining operands (GPRs and FPRs, respectively), allowing integer and floating-point calculations to occur simultaneously without interference.

The core provides support for single-cycle store, and it provides an adder/comparator in the system register unit that allows the dispatch and execution of multiple integer add and compare instructions on each cycle.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing among processor cores varies accordingly.

7.3.7 Core Interface

The core interface is specific for each processor core implementation.

The MPC8360 contains an internal coherent system bus (CSB) that interfaces the processor core to the peripheral logic. This internal bus is very similar in function to the external 60x bus interface on the MPC603e. In the case of the MPC8360, the CSB system logic decodes e300-initiated transactions and directs all accesses to the appropriate interface.

The e300 core can operate at a variety of frequencies allowing the designer to trade off performance for power consumption. The processor core is clocked from a separate PLL, which is referenced to the CSB frequency. This allows the processor core and the peripheral logic to operate at different frequencies.

The e300 core provides a versatile core interface that allows for a wide range of implementations. The interface includes a 32-bit address bus, a 64-bit data bus, and 56 control and information signals (see Figure 7-4). 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.

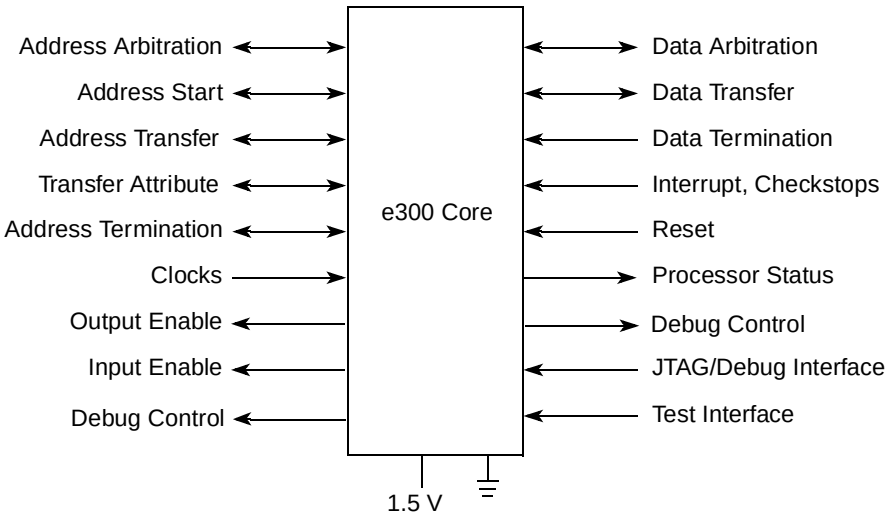


Figure 7-4. Core Interface

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 the IEEE 1149.1 standard) interface and debug unit provides a serial interface to the system for performing monitoring and boundary tests. Two additional signals are added to the e300 core to allow observation of the internal clock state of the core (*stopped*) and to allow the external input to force the core into a halted state (*ext_halt*).
- Core status and control—These signals include the memory reservation signal, machine quiesce control signals, time base 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 MPC8360.
Instruction cancel extension	—	The e300 instruction cancel mechanism improves utilization of instruction cache by supporting ‘hits-under-cancels’ and ‘misses-under-cancels’; the G2_LE requires the cancel to complete before new instruction fetches can begin.
Instruction fetch bursts to caching-inhibited space	Single-beat instruction fetches to caching-inhibited space	The e300’s instruction fetch burst extension allows all caching-inhibited instruction fetches to be performed on the bus as burst transactions, even though the instructions are not cached. This improves performance for instruction space that is caching-inhibited, because up to eight instructions are returned with one bus operation. The G2_LE core must use single-beat instruction fetches for caching-inhibited space, returning only two instructions per bus operation.
Instruction cache way protection	—	The e300 core can protect locked ways in the instruction cache from invalidation; the G2_LE does not support instruction cache way protection.

Table 7-8. Differences Between e300 and G2_LE Cores (continued)

e300 Core	G2_LE Core	Impact
Data cache queue sharing	—	The e300 has a new data cache queue sharing extension that allows the two burst-write queues in the bus unit to be used interchangeably for cache replacements and snoop pushes. Thus, the data cache can support two outstanding cache replacements or two outstanding snoop push operations on the bus at any given time.
icbt instruction	—	The e300 supports a new instruction cache block touch instruction that facilitates preloading the instruction cache before locking; the G2_LE core requires speculatively fetching instructions before locking the instruction cache.
1-1/2-level bus pipelining	1-level bus pipelining	For the e300, a new transaction can complete an address tenure when the previous transaction has been granted the data bus; for the G2_LE, a new transaction must wait until the previous data tenure has completed before completing its address tenure.
PowerPC little-endian not supported	PowerPC little-endian supported	PowerPC little-endian will not be supported in the e300 core, although true little-endian will be fully supported.
Data retry mode removed	Data retry mode available	$\overline{drtry}$ and *drtrymode* will no longer be supported on the e300 and future versions.
External control instructions removed	External control instructions available	The **eciwx** and **ecowx** instruction pair will not be supported on the e300 core. These are optional instructions in the PowerPC architecture.
Reduced pin mode removed	Reduced pin mode available	Reduced pinout mode and the signal *redpinmode* will not be supported in the e300 core.

Chapter 8
Integrated Programmable Interrupt Controller (IPIC)

This chapter describes the integrated programmable interrupt controller (IPIC), including a definition of the external signals and their functions. Also, the configuration, control, and status registers are described in this chapter. Note that individual chapters in this reference manual describe specific initialization aspects for each individual block.

8.1 Introduction

This chapter describes the IPIC interrupt protocol, various types of interrupt sources controlled by the IPIC unit, and the IPIC registers with some programming guidelines. The programming model is similar to the interrupt controller of the MPC8260. The interrupt controller provides interrupt management that is responsible for receiving hardware-generated interrupts from different sources (both internal and external). It also prioritizes and delivers the interrupts to the CPU for servicing. The IPIC prioritizes and manages interrupts from the following controller units:

- Dual DDR memory controllers (DDR): DDR and secondary DDR
- Local bus memory controller (LBC)
- PCI
- Four-channel DMA controller (DMA)
- Message unit (MU)
- DUART communication module (DUART)
- Security engine (SEC)
- System bus arbiter (SBA)
- Periodic interval timer (PIT)
- Real time clock timer (RTC ALR and RTC SEC)
- Eight global timers (GTM1–GTM8)
- Software watchdog timer (WDT)
- I^2C controller (I^2C)
- Power management controller (PMC)
- QUICC Engine block
- External pins ($\overline{\text{IRQ}}$[0:70:15])

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 ($\overline{IRQ}$[0:70:15]).

The interrupt router of the IPIC monitors the outputs of the internal configuration registers. When the priority is highest in one of the received interrupt signals, the IPIC sets the corresponding bit in one of the interrupt pending registers—system internal interrupt pending register (SIPNR)/system external interrupt pending register (SEPNR). If the interrupt is not masked, the IPIC asserts the $\overline{int}$ signal to indicate an interrupt request to the processor. When the processor is running the specific $\overline{int}$, $\overline{cint}$, or $\overline{smi}$ interrupt handler code, the processor must vectorize the external interrupt handler by explicitly (in software) reading the corresponding interrupt vector register (SIVCR, SCVCR or SMVCR). In response to this read, the IPIC unit returns the vector (associated with the interrupt source) to the interrupt handler routine. In addition, the handler can vectorize different branches of interrupt handling.

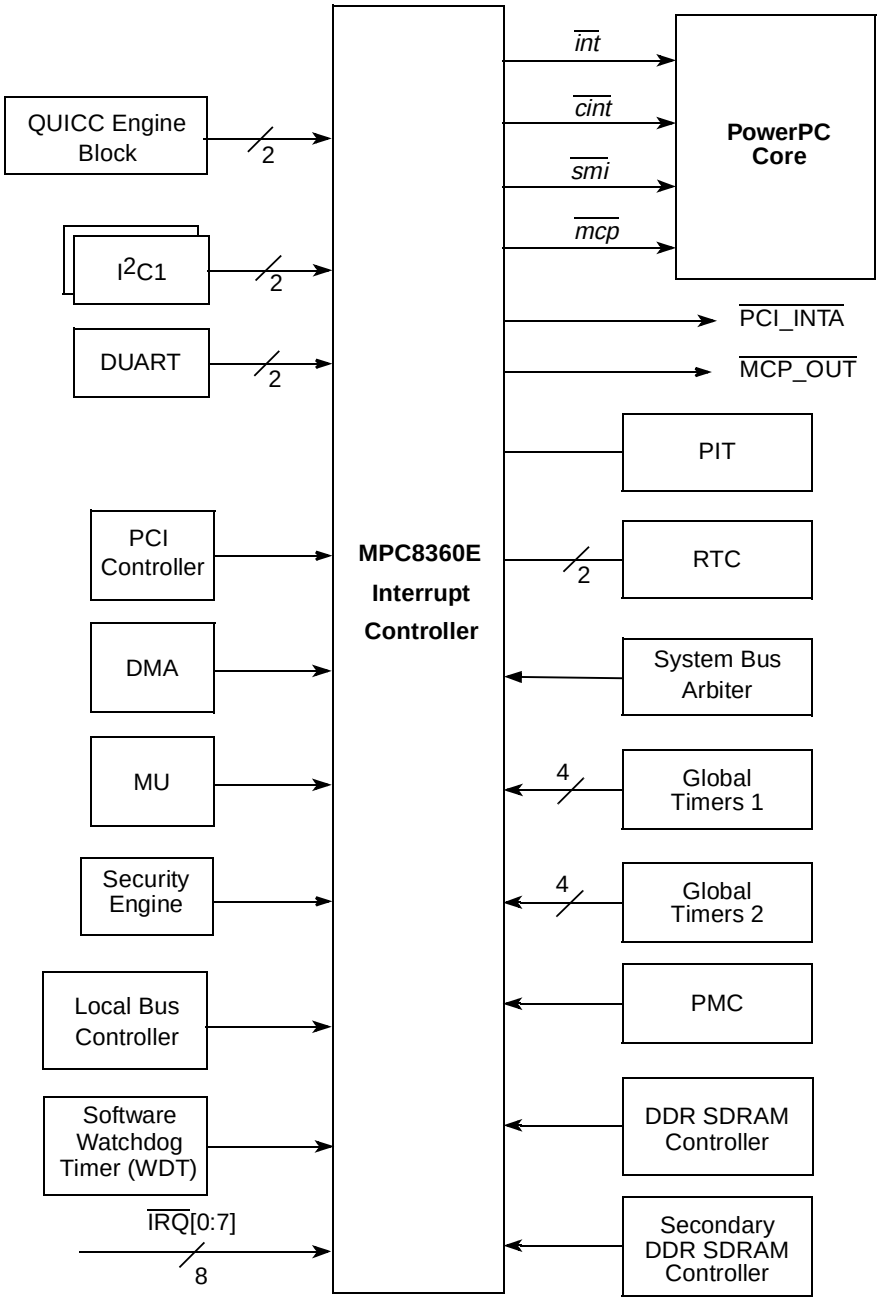


Figure 8-1. Interrupt Sources Block Diagram

The IPIC receives the following types of interrupts:

- External interrupt—triggered by the off-chip signals ($\overline{\text{IRQ}}n$) 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
- TwoFour 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 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 *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 816 distinct external interrupt request input signals (IRQ[0:70:15]) 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[0:70:15]	IRQ[0:70:15]	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[0:70:15]	I	Interrupt request 0–715. 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 (according to the programmed polarity), 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
0x14	Reserved	—	—	—
0x18	Reserved	—	—	—
0x1C	System internal interrupt group D priority register (SIPRR_D)	R/W	0x0530_9770	8.5.5/8-14
0x20	System internal interrupt mask register (SIMSR_H)	R/W	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	8.5.7/8-16
0x2C	System external interrupt pending register (SEPNR)	R/W	Special	8.5.8/8-18
0x30	System mixed interrupt group A priority register (SMPRR_A)	R/W	0x0530_9770	8.5.9/8-18
0x34	System mixed interrupt group B priority register (SMPRR_B)	R/W	0x0530_9770	8.5.10/8-19
0x38	System external interrupt mask register (SEMSR)	R/W	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	8.5.13/8-23
0x44	System error mask register (SERMR)	R/W	0xFF00_00000 xFFFE0000	8.5.14/8-24
0x48	System error control register (SERCR)	R/W	0x0000_0000	8.5.15/8-25
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	8.5.18/8-27
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	8.5.20/8-28
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	8.5.21/8-29
0x10	QUICC Engine ports interrupt mask register (CEPIMR)	R/W	0x0000_0000	8.5.22/8-30
0x14	QUICC Engine ports interrupt control register (CEPICR)	R/W	0x0000_0000	8.5.23/8-31

8.5.1 System Global Interrupt Configuration Register (SICFR)

SICFR, shown in Figure 8-2, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table. See Table 8-5 for more information.

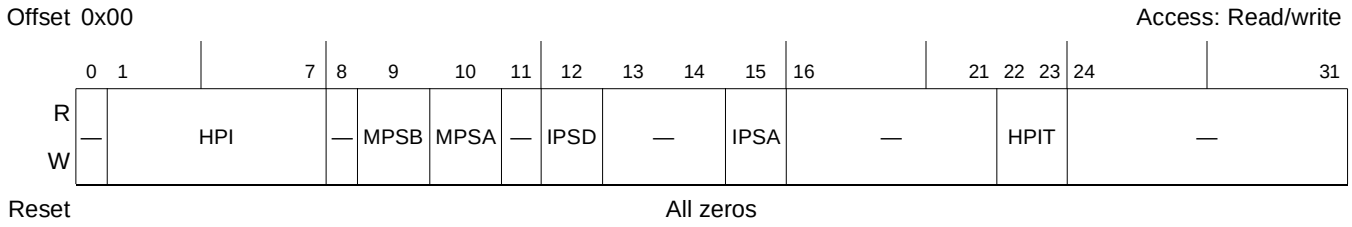


Figure 8-2. System Global Interrupt Configuration Register (SICFR)

Table 8-5 defines the bit fields of SICFR.

Table 8-5. SICFR Field Descriptions

Bits	Name	Description
0	—	Write ignored, read = 0
1–7	HPI	Highest priority interrupt. Specifies the 7-bit unique interrupt number/vector (see Table 8-7) of the single interrupt controller interrupt source that is advanced to the highest priority in the IPIC priority table (see Table 8-35). HPI can be modified dynamically.
8	—	Write ignored, read = 0
9	MPSB	Mixed interrupts priority scheme for group B. Selects the relative MIXB priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXBs are grouped by priority at the top of the table. 1 Spread. The MIXBs are spread by priority in the table.
10	MPSA	Mixed interrupts priority scheme for group A. Selects the relative MIXA priority scheme. It cannot be changed dynamically. 0 Grouped. The MIXAs are grouped by priority at the top of the table. 1 Spread. The MIXAs are spread by priority in the table.
11	—	Write ignored, read = 0
12	IPSD	Internal interrupts priority scheme for group D. Selects the relative SYSD priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSDs are grouped by priority at the top of the table. 1 Spread. The SYSDs are spread by priority in the table.
13–14	—	Write ignored, read = 0
15	IPSA	Internal interrupts priority scheme for group A. Selects the relative SYSA priority scheme. It cannot be changed dynamically. 0 Grouped. The SYSAs are grouped by priority at the top of the table. 1 Spread. The SYSAs are spread by priority in the table.
16–21	—	Write ignored, read = 0

Table 8-5. 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 8-3, 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).

Note that IVECx field specifies the 6 LSBs of IVEC. Will be used for CPM/QUICC Engine IPIC implementation only (case then ipi_int_internal[32:63] signals are not used).

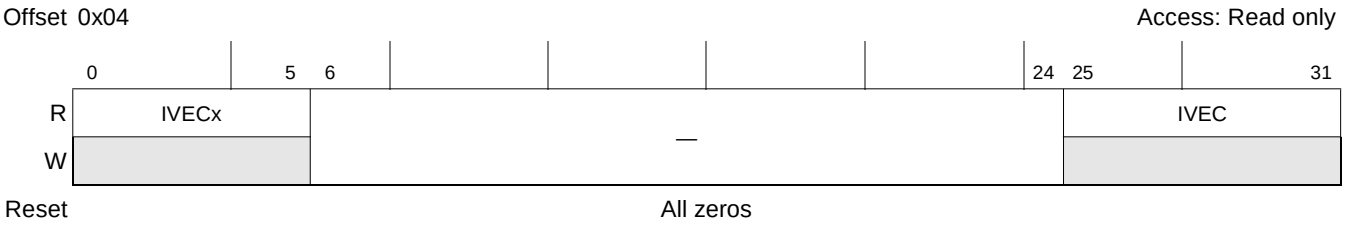


Figure 8-3. System Global Interrupt Vector Register (SIVCR)

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.

Table 8-6. SIVCR Field Descriptions (continued)

Bits	Name	Description
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	I2C1	0b000_1110
15	I2C2	0b000_1111
16	Reserved	0b001_0000
17	IRQ1	0b001_0001
18	IRQ2	0b001_0010
19	IRQ3	0b001_0011
20	IRQ4	0b001_0100
21	IRQ5	0b001_0101
22	IRQ6	0b001_0110
23	IRQ7	0b001_0111
24–31	Reserved	0b001_1000–0b001_1111
32	QUICC Engine High	0b010_0000
33	QUICC Engine Low	0b010_0001
34–47	Reserved	0b010_0010–0b010_1111
48	IRQ0	0b011_0000
49–63	Reserved	0b011_0001–0b011_1111
64	RTC SEC	0b100_0000
65	PIT	0b100_0001
66	PCI	0b100_0010
67	Reserved	0b100_0011

Table 8-7. IVEC/CVEC/MVEC Field Definition (continued)

Interrupt ID Number	Interrupt Meaning	Interrupt Vector
68	RTC ALR	0b100_0100
69	MU	0b100_0101
70	SBA	0b100_0110
71	DMA	0b100_0111
72	GTM4	0b100_1000
73	GTM8	0b100_1001
74	QUICC Engine Ports	0b100_1010
75	SDDR	0b100_1011
76	DDR	0b100_1100
77	LBC	0b100_1101
78	GTM2	0b100_1110
79	GTM6	0b100_1111
80	PMC	0b101_0000
81–83	Reserved	0b101_0001–0b101_0011
84	GTM3	0b101_0100
85	GTM7	0b101_0101
86–89	Reserved	0b101_0110–0b101_1001
90	GTM1	0b101_1010
91	GTM5	0b101_1011
92–127	Reserved	0b101_1100–0b111_1111

8.5.3 System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L)

Each bit in SIPNR_H and SIPNR_L, shown in Figure 8-4 and Figure 8-5, may be assigned an internal interrupt source. (Implemented bits are listed in Table 8-8.) When an interrupt request is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit.

Note that SIPNR bit positions are not changed according to relative priority.

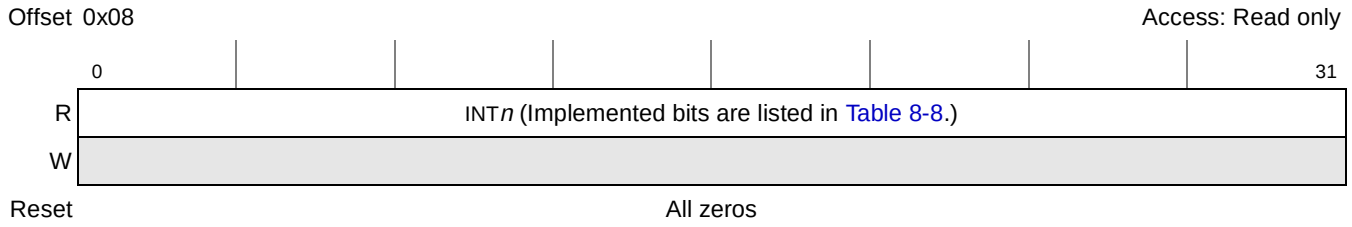


Figure 8-4. System Internal Interrupt Pending Register (SIPNR_H)

Table 8-8 lists implemented SIPNR_H fields. Note that these field descriptions are also valid for SIFCR_H and SIMSR_H.

Table 8-8. SIPNR_H/SIFCR_H/SIMSR_H Bit Assignments

Bits	Field
0	QE High
1	QE Low
2	—
3	—
4	—
5	—
6	—
7	—
8–23	—
24	UART1
25	UART2
26	SEC
27–28	—
29	I2C1
30	I2C2
31	—

Table 8-9 defines the bit fields of SIPNR_H.

Table 8-9. SIPNR_H Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-8) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

SIPNR_L is shown in Figure 8-4.

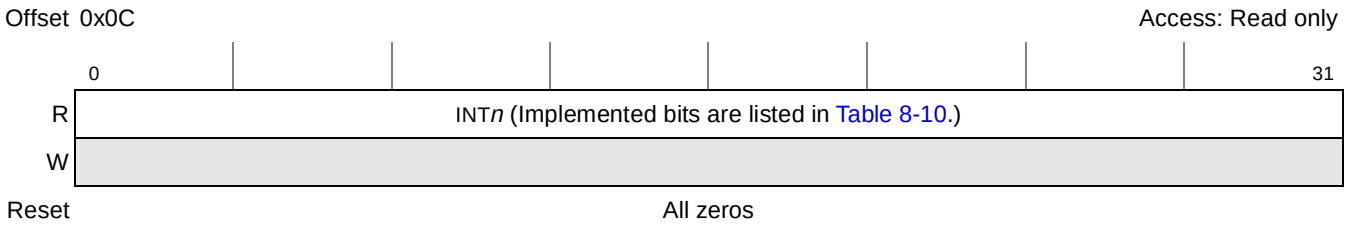


Figure 8-5. System Internal Interrupt Pending Register (SIPNR_L)

Table 8-10 lists implemented SIPNR_L fields. Note that these field assignments are also valid for SIFCR_L and SIMSR_L.

Table 8-10. SIPNR_L/SIFCR_L/SIMSR_L Bit Assignments

Bits	Field
0	RTC SEC
1	PIT
2	PCI
3	—
4	RTC ALR
5	MU
6	SBA
7	DMA
8	GTM4
9	GTM8
10	QE Ports
11	SDDR
12	DDR
13	LBC
14	GTM2
15	GTM6
16	PMC
17–19	—
20	GTM3
21	GTM7
22–25	—
26	GTM1
27	GTM5
28–30	—
31	—

Table 8-11 defines the bit fields of SIPNR_L.

Table 8-11. SIPNR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit (listed in Table 8-10) corresponds to an internal interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit. When a pending interrupt is handled, the user clears the SIPNR bit by clearing the corresponding event register bit. SIPNR bits are read only. Writing to this register has no effect. Note that the SIPNR bit positions are not changed according to their relative priority. For unimplemented bits, writes are ignored, read = 0.

8.5.4 System Internal Interrupt Group A Priority Register (SIPRR_A)

The system internal interrupt group A priority register (SIPRR_A), shown in Figure 8-6, defines the priority between QE High and QE Low internal interrupt signals.

Offset 0x10 Access: Read/write

	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	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 A Priority Register (SIPRR_A)

Table 8-12 defines the bit fields of SIPRR_A.

Table 8-12. SIPRR_A Field Descriptions

Bits	Name	Description
0–2	SYSA0P	SYSA0 priority order. Defines which interrupt source asserts its request in the SYSA0 priority position. The user should not program the same code to multiple priority positions (0–7). These bits can be changed dynamically. The definition of SYSA0P is as follows: 000 QE High asserts its request in the SYSA0 position. 001 QE Low asserts its request in the SYSA0 position. 010 Reserved 011 Reserved 100 Reserved 101 Reserved 110 Reserved 111 Reserved
3–11, 16–27	SYSA1P–SYSA7P	Same as SYSA0P, but for SYSA1P–SYSA7P.
12–15, 28–31	—	Write ignored, read = 0

8.5.5 System Internal Interrupt Group D Priority Register (SIPRR_D)

SIPRR_D, shown in Figure 8-7, defines the priority among the interrupt sources listed in Table 8-13.

Offset 0x1C Access: Read/write

	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	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 8-7. 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 I2C1 asserts its request in the SYSD0 position. 110 I2C2 asserts its request in the SYSD0 position. 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 8-8 and Figure 9-9, 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.

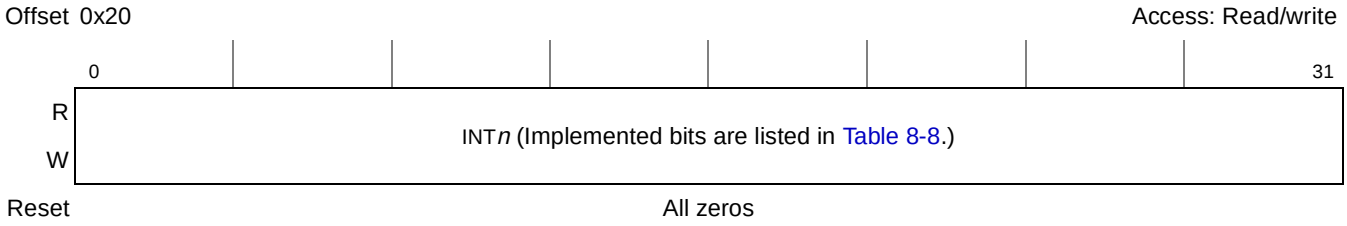


Figure 8-8. System Internal Interrupt Mask Register (SIMSR_H)

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.

SIMSR_L is shown in Figure 8-9.

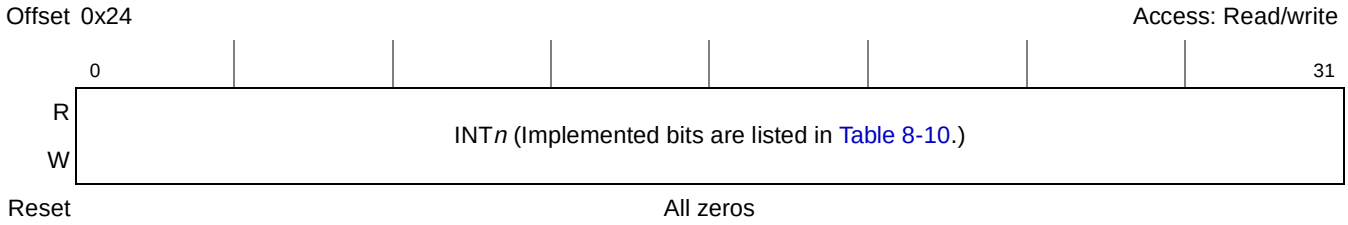


Figure 8-9. System Internal Interrupt Mask Register (SIMSR_L)

Table 8-15 defines the bit fields of SIMSR_L.

Table 8-15. SIMSR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	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 8-9, are ignored on writes; read = 0.

8.5.7 System Internal Interrupt Control Register (SICNR)

SICNR, shown in Figure 8-10, defines the IPIC output interrupt type ($\overline{\text{int}}$, $\overline{\text{cint}}$, or $\overline{\text{smi}}$) in the SYSA0–SYSA1 and SYSD0–SYSD1 priority positions. All other priority positions assert $\overline{\text{int}}$ to the core.

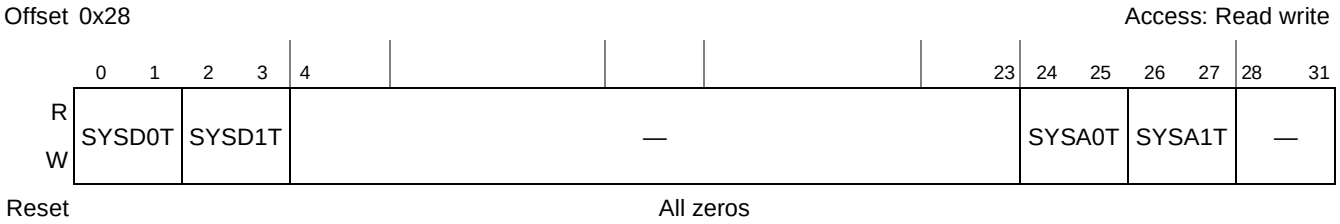


Figure 8-10. System Internal Interrupt Control Register (SICNR)

Table 8-16 defines the bit fields of SICNR.

Table 8-16. SICNR Field Descriptions

Bits	Name	Description
0–1	SYSD0T	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 8-11, 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	9	10	11	12	13	14	15
R	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	—							
W																
R	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
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 8-11. System External Interrupt Pending Register (SEPNR)

Table 8-17 defines the bit fields of SEPNR.

Table 8-17. SEPNR Field Descriptions

Bits	Name	Description
0–715	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.
816–31	—	Write ignored, read = 0

8.5.9 System Mixed Interrupt Group A Priority Register (SMPRR_A)

The SMPRR_A, shown in Figure 8-12, 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 W	MIXA0P	MIXA1P	MIXA2P	MIXA3P	—	MIXA4P	MIXA5P	MIXA6P	MIXA7P	—
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 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 8-13, defines the priority among the sources listed in Table 8-19.

Offset 0x34 Access: Read/write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R 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 8-13. System Mixed Interrupt Group B Priority Register (SMPRR_B)

Table 8-19 defines the bit fields of SMPRR_B.

Table 8-19. SMPRR_B Field Descriptions

Bits	Name	Description
0–2 3–11, 16–27	MIXB*n*P	MIXB*n* priority order. Defines which interrupt source asserts its request in the MIXB*n* priority position. The user must not program the same code to more than one priority position (0–7). These bits can be changed dynamically. The definition of MIXB*n*P is as follows: 000 RTC ALR asserts its request to the MIXB*n* position. 001 MU asserts its request to the MIXB*n* position. 010 SBA asserts its request to the MIXB*n* position. 011 DMA asserts its request to the MIXB*n* position. 100 IRQ4 asserts its request to the MIXB*n* position. 101 IRQ5 asserts its request to the MIXB*n* position. 110 IRQ6 asserts its request to the MIXB*n* position. 111 IRQ7 asserts its request to the MIXB*n* position.
12–15, 28–31	—	Write ignored, read = 0

8.5.11 System External Interrupt Mask Register (SEMSR)

Each bit in the system external interrupt mask register (SEMSR), shown in Figure 8-11, corresponds to an external interrupt source. The user masks an interrupt by clearing the corresponding SEMSR bit. An interrupt is unmasked (enabled) by setting the corresponding SEMSR bit.

When an external interrupt request occurs, the corresponding 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	6	7	8	9	10	11	12	13	14	15
R W	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	—							
R W	IRQ0[1]	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5	IRQ6	IRQ7	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
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 8-14. System External Interrupt Mask Register (SEMSR)

Table 8-20 defines the bit fields of SEMSR.

Table 8-20. SEMSR Field Descriptions

Bits	Name	Description
0–715	—	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.
816–15	—	Write ignored, read = 0
16	SIRQ0	Steer IRQ0. 0 IRQ0 is used as external interrupt request 1 IRQ0 is used as external MCP request
17–31	—	Write ignored, read = 0

8.5.12 System External Interrupt Control Register (SECNR)

SECNR, shown in Figure 8-15, defines the edge detect mode for external $\overline{\text{IRQ}}n$ 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.

Offset 0x3C Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIXB0T		MIXB1T		—				MIXA0T		MIXA1T		—			
R W	MIXB0T		MIXB1T		—				MIXA0T		MIXA1T		—			
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EDI0	EDI1	EDI2	EDI3	EDI4	EDI5	EDI6	EDI7	—							
R W	EDI0	EDI1	EDI2	EDI3	EDI4	EDI5	EDI6	EDI7	EDI8	EDI9	EDI10	EDI11	EDI12	EDI13	EDI14	EDI15
Reset	All zeros															

Figure 8-15. System External Interrupt Control Register (SECNR)

Table 8-21 defines the bit fields of SECNR.

Table 8-21. SECNR Field Descriptions

Bits	Name	Description
0–1	MIXB0T	MIXB0 priority position IPIC output interrupt type. Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the MIXB0 priority position. These bits can be changed dynamically. The definition of MIXB0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXB0. 01 $\overline{smi}$ request is asserted to the core for MIXB0. 10 $\overline{cint}$ request is asserted to the core for MIXB0. 11 Reserved
2–3	MIXB1T	Same as MIXB0T, but for MIXB1T.
4–7	—	Write ignored, read = 0
8–9	MIXA0T	MIXA0 priority position IPIC output interrupt Type. Defines which type of the IPIC output interrupt signal ($\overline{int}$, $\overline{cint}$, or $\overline{smi}$) asserts its request to the core in the MIXA0 priority position. These bits can be changed dynamically. The definition of MIXA0T is as follows: 00 $\overline{int}$ request is asserted to the core for MIXA0. 01 $\overline{smi}$ request is asserted to the core for MIXA0. 10 $\overline{cint}$ request is asserted to the core for MIXA0. 11 Reserved
10–11	MIXA1T	Same as MIXA0T, but for MIXA1T.
12–15	—	Write ignored, read = 0

Table 8-21. SECNR Field Descriptions (continued)

Bits	Name	Description
16–2331	EDIx	Each bit defines the edge detect mode for the external $\overline{\text{IRQ}}n$ interrupt signals, determines whether the corresponding $\overline{\text{IRQ}}n$ signal asserts an interrupt request upon either a high-to-low change or low assertion on the pin. The corresponding $\overline{\text{IRQ}}n$ signal asserts an interrupt request as follows: 0 Low assertion on $\overline{\text{IRQ}}n$ generates an interrupt request (level sensitive). 1 High-to-low change on $\overline{\text{IRQ}}n$ 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 8-16, correspond to the external and internal non-maskable error source machine check (mcp) conditions listed in Table 8-22. When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit.

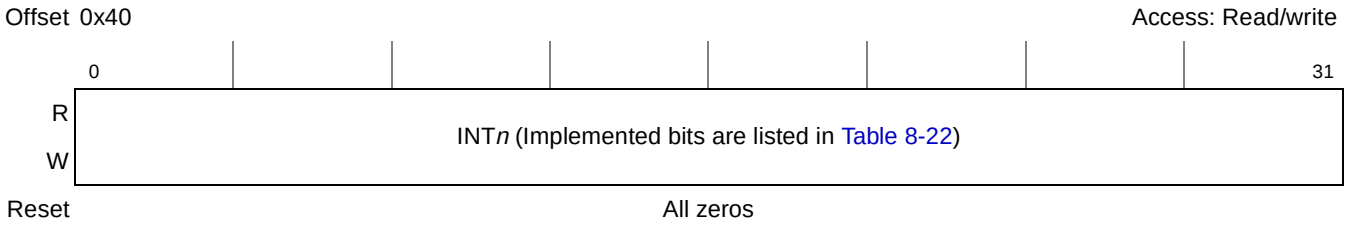


Figure 8-16. System Error Status Register (SERSR)

Table 8-22 lists the implemented SERSR bits. Note that these field assignments are valid for SERMR and SERFR.

Table 8-22. SERSR/SERMR/SERFR Bit Assignments

Bits	Field
0	IRQ0[1]
1	WDT
2	SBA
3	CIEE
4	CMEE
5	PCI
6	—
7	MU
8–14	—
15	—
16–31	—

[1] This bit is valid only if the IRQ0 signal is configured as an external MCP interrupt (SEMSR[SIRQ0] = 1)

Table 8-23 defines the bit fields of SERSR.

Table 8-23. SERSR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit in the SERSR, listed in Table 8-22, corresponds to an external and an internal error source (mcp). When an error interrupt signal is received, the interrupt controller sets the corresponding SERSR bit. SERSR bits are cleared by writing ones to them. Unmasked event register bits should be cleared before clearing SERSR bits. Because the user can only clear bits in this register, writing zeros to this register has no effect. SERSR bits are cleared by power-on reset. Subsequent soft and hard resets do not affect SERSR bit states. For unimplemented bits (listed as reserved in Table 8-22), writes are ignored, read = 0

8.5.14 System Error Mask Register (SERMR)

Each implemented bit in SERMR, shown in Figure 8-17, corresponds to an external and an internal $\overline{mcp}$ source (MCP). The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the corresponding SERMR bit although no MCP request is passed to the core in this case. The SERMR can be read by the user at any time.

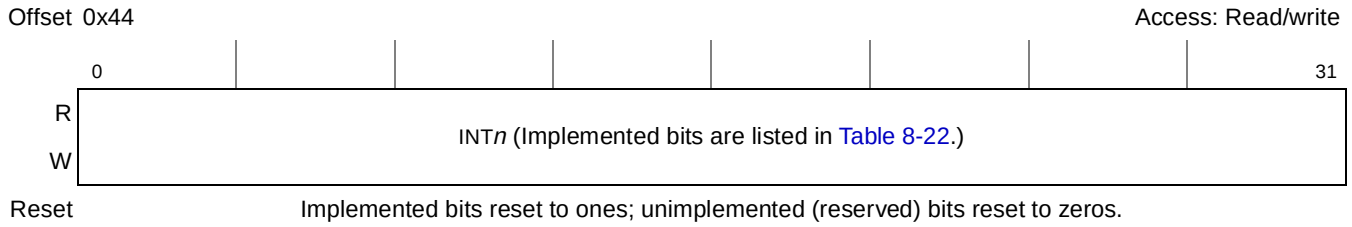


Figure 8-17. System Error Mask Register (SERMR)

Table 8-24 defines the bit fields of SERMR.

Table 8-24. SERMR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented SERMR bit, listed in Table 8-22, corresponds to an external and an internal MCP source. The user masks an MCP by clearing and enables an interrupt by setting the corresponding SERMR bit. When a masked MCP occurs, the corresponding SERSR bit is set, regardless of the setting of the SERMR bit although no MCP request is passed to the core. The SERMR can be read by the user at any time. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.15 System Error Control Register (SERCR)

SERCR, shown in Figure 8-18, defines the control bits that route MCP requests in core disable mode to either MCP_OUT or PCI_INTA in core-disable mode.

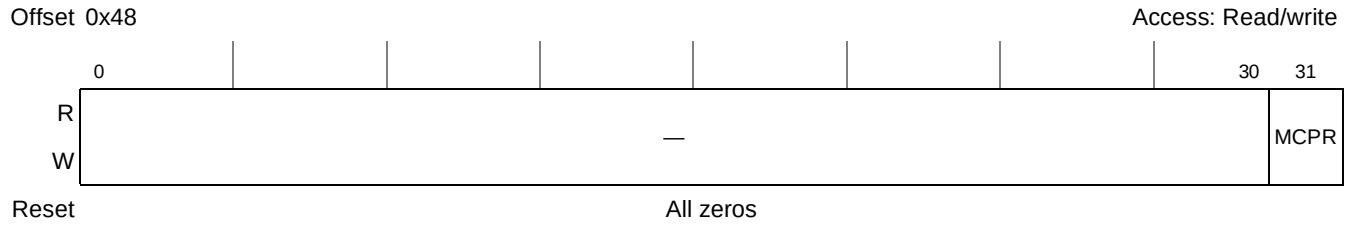


Figure 8-18. System Error Control Register (SERCR)

Table 8-25 defines the bit fields of SERCR.

Table 8-25. SERCR Field Descriptions

Bits	Name	Description
0–30	—	Write ignored, read = 0
31	MCPR	MCP route. Route MCP request to either MCP_OUT or PCI_INTA (in core disable mode). 0 MCP routed to PCI_INTA (in core disable mode). 1 MCP routed to MCP_OUT (in core disable mode).

8.5.16 System Internal Interrupt Force Registers (SIFCR_H and SIFCR_L)

Each implemented bit SIFCR_H and SIFCR_L, shown in Figure 8-19 and Figure 8-20, corresponds to an internal interrupt source. When a bit is set, the interrupt controller generates the corresponding interrupt (sets the corresponding SIPNR bit).The SIFCR can be read by the user at any time.

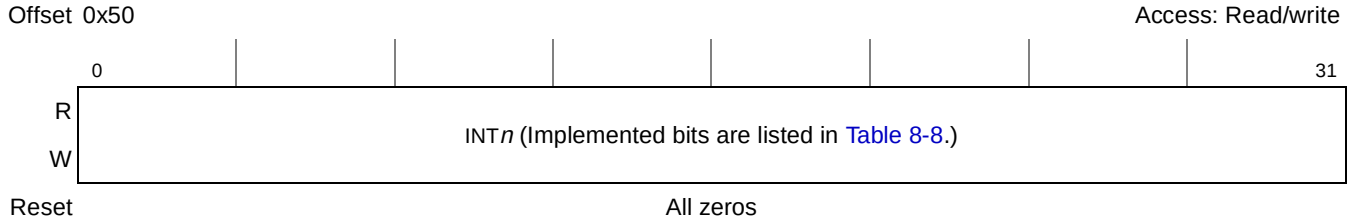


Figure 8-19. System Internal Interrupt Force Register (SIFCR_H)

Table 8-26 defines the bit fields of SIFCR_H.

Table 8-26. SIFCR_H Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-8, corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR*x* bit. SIFCR*n* bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

SIFCR_L is shown in Figure 8-20.

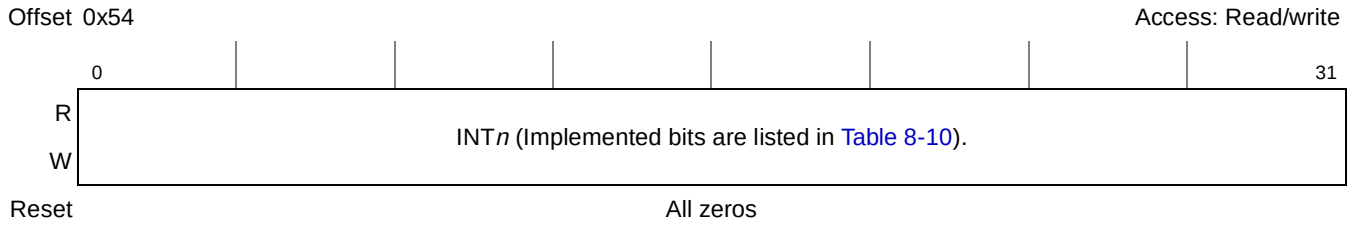


Figure 8-20. System Internal Interrupt Force Register (SIFCR_L)

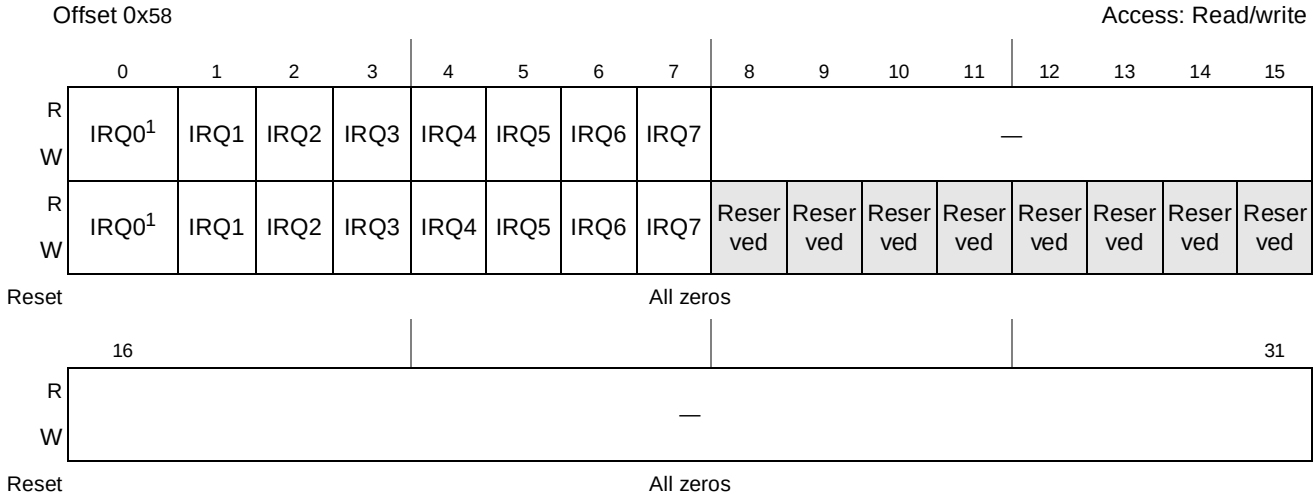
Table 8-27 defines the bit fields of SIFCR_L.

Table 8-27. SIFCR_L Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-10, corresponds to an internal interrupt source. The user forces an interrupt by setting the corresponding SIFCR*x* bit. SIFCR*x* bit positions are not changed according to their relative priority. Writes to unimplemented (reserved) bits are ignored; read = 0

8.5.17 System External Interrupt Force Register (SEFCR)

Each implemented bit in SEFCR, shown in Figure 8-21, corresponds to an external interrupt source. When a bit is set, the interrupt controller generates the corresponding external interrupt (sets the corresponding SEPNR bit). SEFCR can be read by the user at any time.



1 This bit is valid only if IRQ0 is configured as an external maskable interrupt (SEMSR[SIRQ0] = 0)

Figure 8-21. System External Interrupt Force Register (SEFCR)

Table 8-28 defines the bit fields of SEFCR.

Table 8-28. SEFCR Field Descriptions

Bits	Name	Description
0–715	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.
816–31	—	Write ignored, read = 0

8.5.18 System Error Force Register (SERFR)

Each bit in the system error force register (SERFR), shown in Figure 8-22, corresponds to an external MCP source. When a bit is set, the interrupt controller generates the corresponding MCP interrupt (sets the corresponding SERSR bit). The SERFR can be read by the user at any time.

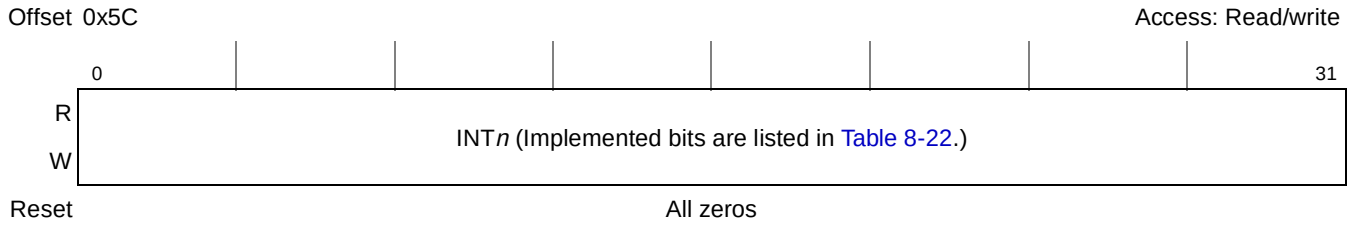


Figure 8-22. System Error Status Register (SERFR)

Table 8-29 defines the bit fields of SERFR.

Table 8-29. SERFR Field Descriptions

Bits	Name	Description
0–31	INT*n*	Each implemented bit, listed in Table 8-22, corresponds to an external MCP source. The user forces an MCP by setting the SERFR bit. SERFR bit positions are not affected by their relative priority. Attempts to write to unimplemented (reserved) bits are ignored; read = 0

8.5.19 System Critical Interrupt Vector Register (SCVCR)

SCVCR, shown in Figure 8-23, contains a 7-bit code (Table 8-30) representing the unmasked critical interrupt ($\overline{\text{CINT}}$) 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).

Note that the CVECx field specifies the 6 LSBs of CVEC. Will be used for CPM/QUICC Engine IPIC implementation only (case then ipi_int_internal[32:63] signals are not used).

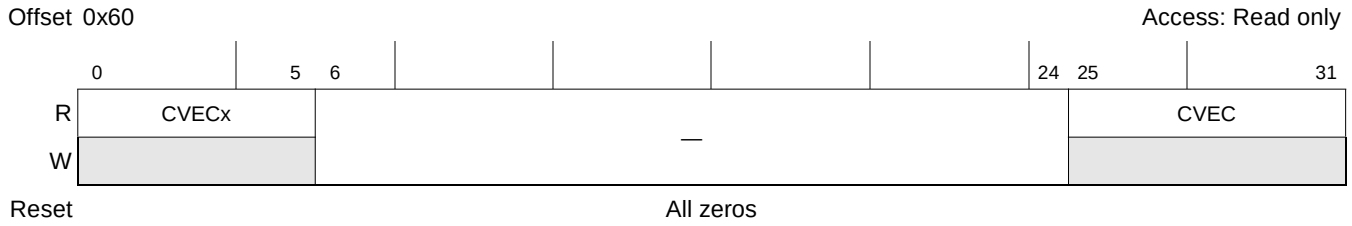


Figure 8-23. System Critical Interrupt Vector Register (SCVCR)

Table 8-30 defines SCVCR bit fields.

Table 8-30. SCVCR Field Descriptions

Bits	Name	Description
0–5	CVECx	Backward (MPC8260) compatible critical interrupt vector. Specifies a 6-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVECx field will correctly reflect only first 64 interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.
6–24	—	Write ignored, read = 0
25–31	CVEC	Critical interrupt vector. Specifies a 7-bit unique number of the IPIC's highest priority critical interrupt source, pending to the core. When a critical interrupt request occurs, SCVCR can be read. If there are multiple critical interrupt sources, SCVCR latches the highest priority critical interrupt. Note that CVEC field correctly reflects all of the interrupt vectors (See Table 8-7 for details). The value of SCVEC cannot change while it is being read.

8.5.20 System Management Interrupt Vector Register (SMVCR)

SMVCR, shown in Figure 8-24, contains a 7-bit code (Table 8-31) representing the unmasked system management interrupt ($\overline{\text{SMI}}$) 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).

Note that MVECx field specifies the 6 LSBs of MVEC. Will be used for CPM/QUICC Engine IPIC implementation only (case then ipi_int_internal[32:63] signals are not used).

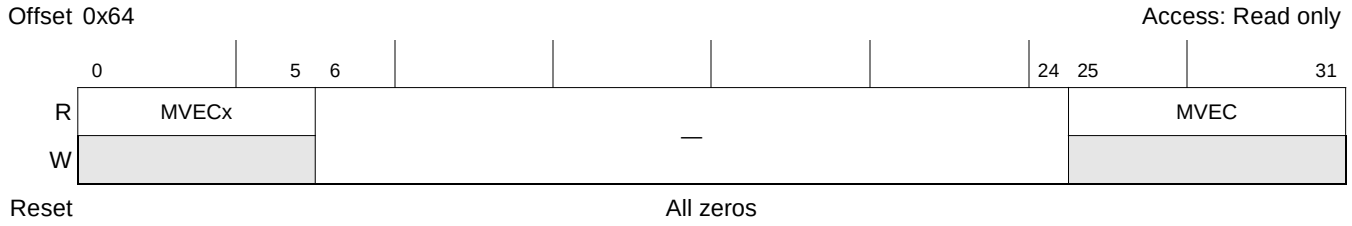


Figure 8-24. System Management Interrupt Vector Register (SMVCR)

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 8-25, 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	11	12	13	14	15
R	PA[15]	PA[16]	PA[29]	PA[30]	PB[3]	PB[5]	PB[12]	PB[13]	PB[26]	PB[27]	PC[27]	PC[28]	PC[29]	PD[12]	PD[13]	PD[16]
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	Undefined (the user should write 1s to clear before using)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
R	PD[17]	PD[26]	PD[27]	PE[12]	PE[13]	PE[24]	PE[25]	PE[26]	PE[27]	PE[31]	PF[20]	PG[31]	—	
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		
Reset	Undefined (the user should write 1s to clear before using)													

Figure 8-25. QUICC Engine Ports Interrupt Event Register (CEPIER)

[1] Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, "Memory Map."

Table 8-32 defines the bit fields of CEPIER.

Table 8-32. CEPIER Bit Settings

Bits	Name	Description
0–31	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.

8.5.22 QUICC Engine Ports Interrupt Mask Register (CEPIMR)

The QUICC Engine ports interrupt mask register (CEPIMR), shown in Figure 8-26, defines the interrupt masking for the individual QUICC Engine ports lines. When a masked interrupt request occurs, the corresponding CEPIER bit is set, regardless of the CEPIMR state. When one or more non-masked interrupt events occur, the ‘QE Ports’ internal event is generated. The ‘QE Ports’ internal event is one source in the SIPNR register (See Section 8.5.3, “System Internal Interrupt Pending Registers (SIPNR_H and SIPNR_L).”

Offset 0x10[1] Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PA15	PA16	PA29	PA30	PB3	PB5	PB12	PB13	PB26	PB27	PC27	PC28	PC29	PD12	PD13	PD16
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
R W	PD17	PD26	PD27	PE12	PE13	PE24	PE25	PE26	PE27	PE31	PF20	PG31	—	
Reset	All zeros													

Figure 8-26. QUICC Engine Ports Interrupt Mask Register (CEPIMR)

[1] Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, “Memory Map.”

Table 8-33 defines the bit fields of CEPIMR.

Table 8-33. CEPIMR Bit Settings

Bits	Name	Description
0–9	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 8-27, determines whether the corresponding QUICC Engine port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

Offset 0x14[1] Access: Read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PA15	PA16	PA29	PA30	PB3	PB5	PB12	PB13	PB26	PB27	PC27	PC28	PC29	PD12	PD13	PD16
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
R W	PD17	PD26	PD27	PE12	PE13	PE24	PE25	PE26	PE27	PE31	PF20	PG31	—	
Reset	All zeros													

Figure 8-27. QUICC Engine Ports Interrupt Control Register (CEPICR)

[1] Note that the base address for CEPIER is not the same as the base address of other registers in the IPIC. Instead, this register uses a base address which is specific to QUICC engine ports interrupts registers. See Chapter 2, "Memory Map."

Table 8-34 defines the bit fields of CEPICR.

Table 8-34. CEPICR Bit Settings

Bits	Name	Description
0–27	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.
28–31	—	Reserved. Should be cleared.

8.6 Functional Description

The following sections describe the types of interrupts, interrupt configurations, and their priorities.

8.6.1 Interrupt Types

The IPIC is responsible for receiving hardware-generated interrupts from different sources (both internal and external) along with prioritizing and delivering them to the CPU for servicing. The interrupt sources are controlled by the IPIC unit and may cause three types of exceptions in the processor core. The $\overline{int}$ signal is the main interrupt output from the IPIC to the processor core and causes the external interrupt exception.The $\overline{cint}$ signal is the critical interrupt output from the IPIC to the processor core and causes the critical external interrupt exception. The $\overline{smi}$ signal is the system management interrupt output from the IPIC to the processor core and causes the system management interrupt exception. The machine check exception is caused by the internal $\overline{mcp}$ signal generated by the IPIC, informing the processor of error conditions, assertion of the external MCP request, and other conditions.

8.6.2 Interrupt Configuration

Figure 8-28 shows the interrupt configuration.

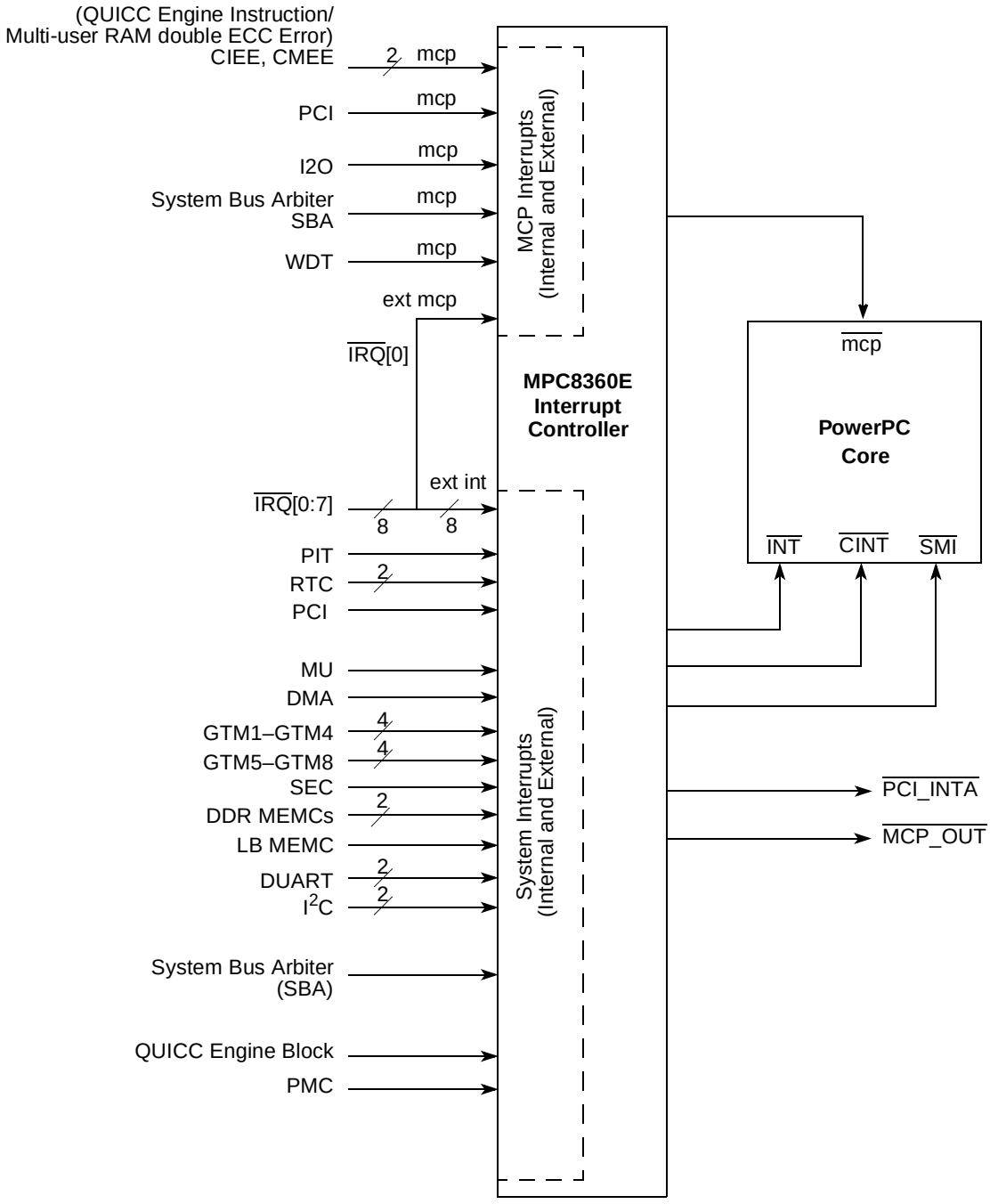


Figure 8-28. Interrupt Structure

The interrupt controller allows masking of each interrupt source. When an unmasked interrupt source is pending in the SIPNR register, the interrupt controller sends an interrupt request to the core. When an

interrupt is taken, the interrupt mask bit in the machine state register is cleared to disable further interrupt requests to the PowerPC core until software can handle them.

All interrupt sources are prioritized and bits are set in the system interrupt pending register (SIPNR, SEPNR) as interrupts occur regardless of whether they are masked in the IPIC. The prioritization of the interrupt sources is flexible within the following groups:

- The relative priority of the QE High and QE Low internal interrupt signals can be modified.
- The relative priority of the Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, and Reserved internal interrupt signals can be modified.
- The relative priority of the Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, Reserved, and Reserved internal interrupt signals can be modified.
- The relative priority of the UART1, UART2, Reserved, Reserved, I2C1, I2C2, and SEC internal interrupt signals can be modified.
- The relative priority of the IRQ0, IRQ1, IRQ2, and IRQ3 external interrupts, and RTC SEC internal interrupts can be modified.
- The relative priority of the IRQ4, IRQ5, IRQ6, and IRQ7 external interrupts, and RTC ALR, MU, SBA, and DMA internal interrupts can be modified.
- One interrupt source can be assigned to be the programmable highest priority.

The SIVCR is updated with a 7-bit vector corresponding to the sub-block with the highest current priority.

8.6.3 Internal Interrupts Group Relative Priority

The relative priority in each internal group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC internal interrupt priority registers (SIPRR*x*) and can be changed dynamically to implement a rotating priority.

In addition, the grouping of the locations of the interrupt entries has the following two options:

- Grouped. In the group scheme, all interrupts are grouped together at the top of Table 8-35, ahead of most other interrupt sources. This scheme is ideal for applications where all interrupt sources function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over Table 8-35 so other sources can have lower interrupt latencies. This scheme is also programmed but cannot be changed dynamically.

8.6.4 Mixed Interrupts Group Relative Priority

The relative priority between up to four internal and four external interrupts in each group is programmable and can be changed dynamically. The group priorities are programmed in the IPIC mixed interrupt priority registers (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)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
27	SYSA5 (Grouped)
28	SYSA6 (Grouped)
29	SYSA7 (Grouped)
30	MIXA4 (Grouped)
31	MIXA5 (Grouped)
32	MIXA6 (Grouped)
33	MIXA7 (Grouped)
34	MIXB2 (Spread)
35–38	Reserved
39	MIXA3 (Spread)
40–43	Reserved
44	MIXB4 (Grouped)
45	MIXB5 (Grouped)
46	MIXB6 (Grouped)
47	MIXB7 (Grouped)
48	MIXB3 (Spread)
49	SYSD0 (Grouped)
50	SYSD1 (Grouped)
51	SYSD2 (Grouped)
52	SYSD3 (Grouped)
53	MIXA4 (Spread)
54	SYSD4 (Grouped)
55	SYSD5 (Grouped)
56	SYSD6 (Grouped)
57	SYSD7 (Grouped)
58	MIXB4 (Spread)
59	GTM4
60	Reserved
61	SYSA0 (Spread)
62	GTM8
63	Reserved
64	SYSD0 (Spread)
65	Reserved
66	QE Ports
67	MIXA5 (Spread)
68	SDDR

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
69	Reserved
70	SYSA1 (Spread)
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)

Table 8-35. Interrupt Source Priority Levels (continued)

Priority	Interrupt Source Description
104	Reserved
105	Reserved
106	SYSA5 (Spread)
107	Reserved
108	Reserved
109	SYSD5 (Spread)
110	Reserved
111	Reserved
112	MIXB7 (Spread)
113	GTM1
114	Reserved
115	SYSA6 (Spread)
116	GTM5
117	Reserved
118	SYSD6 (Spread)
119	Reserved
120	Reserved
121	Reserved
122	Reserved
123	SYSA7 (Spread)
124	Reserved
125	Reserved
126	SYSD7 (Spread)
127	Reserved
128	Reserved

8.6.7 Masking Interrupt Sources

By programming the system interrupt mask registers, SIMSR*x* and SEMSR, the user can mask interrupt requests to the core. Each SIMSR*x* and SEMSR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding SIMSR or SEMSR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIPNR*x* or SEMSR bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

When an interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that particular block. Table 8-35 shows which interrupt sources have multiple interrupting events.

Figure 8-29 shows an example of how the masking occurs, using a DDR block as an example.

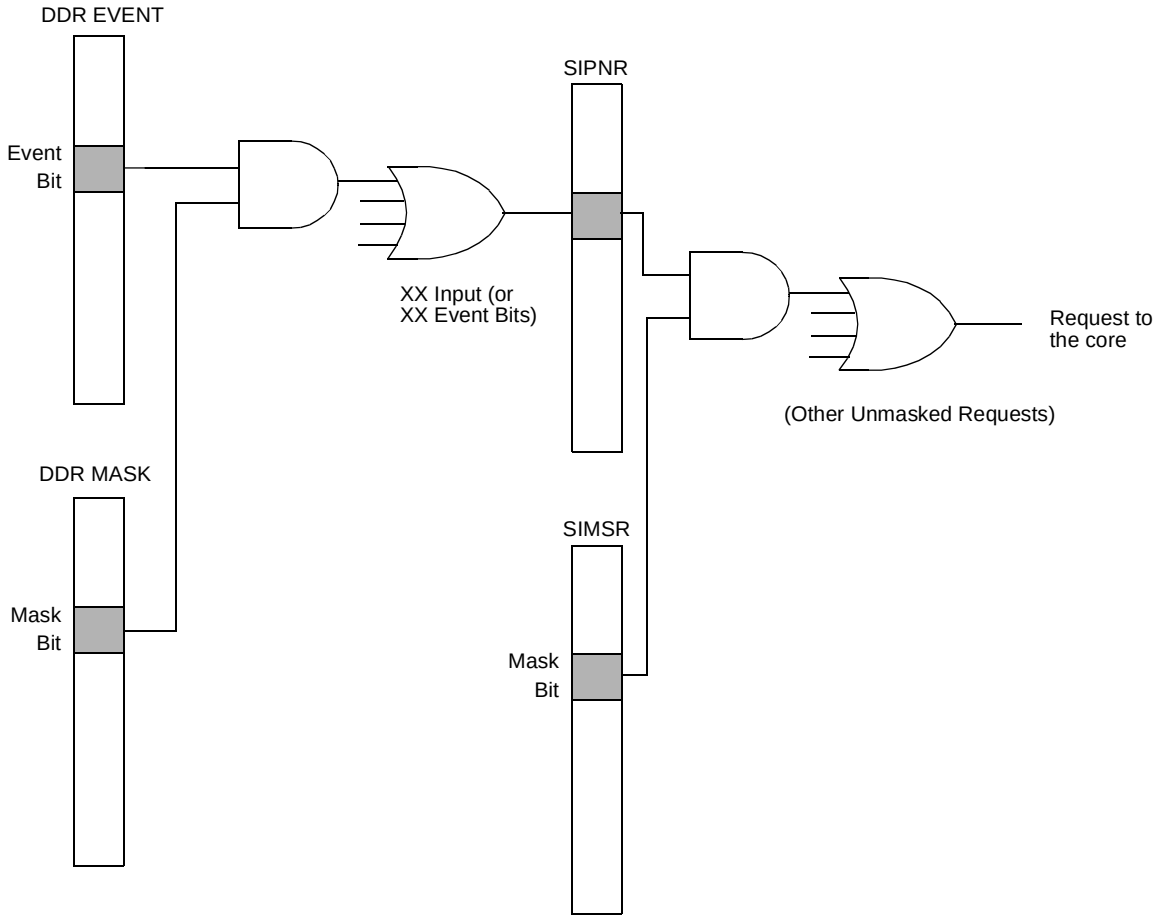


Figure 8-29. DDR Interrupt Request Masking

8.6.8 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority according to Table 8-35. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by interrupt handler software reading SIVCR. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt in response to a read of SIVCR. Table 8-6 lists the encodings for the seven low-order bits of the interrupt vector.

8.6.9 Machine Check Interrupts

The PIC supports the non-maskable machine check interrupts. When an error interrupt signal is received, the interrupt controller indicates the source by setting the corresponding SERSR bit. These sources are listed in Table 8-22.

8.6.10 QUICC Engine Ports Interrupts

The QUICC Engine ports interrupts are a special case of external interrupt requests, which are specifically related to the QUICC Engine block. Each such QUICC Engine ports external interrupt may be generated upon detection of a high-to-low change on the external signal or upon any change on the external signal (note that this is not the same as for normal $\overline{\text{IRQ}}$ 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[15]	UCC1:$\overline{\text{CTS}}$	PD[13]	UCC5:$\overline{\text{CD}}$
PA[16]	UCC1:$\overline{\text{CD}}$	PD[16]	—
PA[29]	UCC2:$\overline{\text{CTS}}$	PD[17]	—
PA[30]	UCC2:$\overline{\text{CD}}$	PD[26]	UCC6:$\overline{\text{CTS}}$
PB[3]	UCC4:$\overline{\text{CTS}}$	PD[27]	UCC6:$\overline{\text{CD}}$
PB[5]	UCC4:$\overline{\text{CD}}$	PE[12]	UCC7:$\overline{\text{CTS}}$
PB[12]	UCC3:$\overline{\text{CTS}}$	PE[13]	UCC7:$\overline{\text{CD}}$
PB[13]	UCC3:$\overline{\text{CD}}$	PE[24]	UCC5:$\overline{\text{CTS}}$
PB[26]	UCC4:$\overline{\text{CTS}}$	PE[25]	UCC5:$\overline{\text{CD}}$
PB[27]	UCC4:$\overline{\text{CD}}$	PE[26]	UCC8:$\overline{\text{CTS}}$
PC[27]	—	PE[27]	UCC8:$\overline{\text{CD}}$
PC[28]	—	PE[31]	—
PC[29]	—	PF[20]	UCC2:$\overline{\text{CD}}$
PD[12]	UCC5:$\overline{\text{CTS}}$	PG[31]	UCC2:$\overline{\text{CTS}}$

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 DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

NOTE

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

The MPC8360 has two DDR memory controllers. Each controller supports a 32-bit data bus and two chip selects (physical banks); however, if only the primary controller is used, a 64-bit bus is supported along with four chip selects. This chapter primarily describes a configuration where one controller is used with four chip selects and a 64-bit data bus.

If both controllers are to be used, they must be set to 32-bit bus mode. This is done by setting DDR_SDRAM_CFG[32_BE] for both controllers. Note that when using DDR1 in 32-bit mode, burst length should be set to 8 beats. This can be done by setting DDR_SDRAM_CFG[8_BE] (for both controllers).

Also note that if 32-bit mode is selected in the primary DDR memory controller, the higher-order data bits (MDQ[32:63]) are used for the secondary DDR memory controller. Other than this, all of the board connections and register programming are identical to 64-bit bus mode.

This chapter also describes a single set of external signals. The signals of the two controllers of the MPC8360E are differentiated elsewhere in this document by means of prefixes. For example, the signal that appears in this chapter as MBA1 may appear elsewhere in this document as either MEMC1_MBA1 or MEMC2_MBA1.

Figure 9-1 is a high-level block diagram of the DDR memory controller with its associated interfaces. Section 9.5, “Functional Description,” contains detailed figures of the controller.

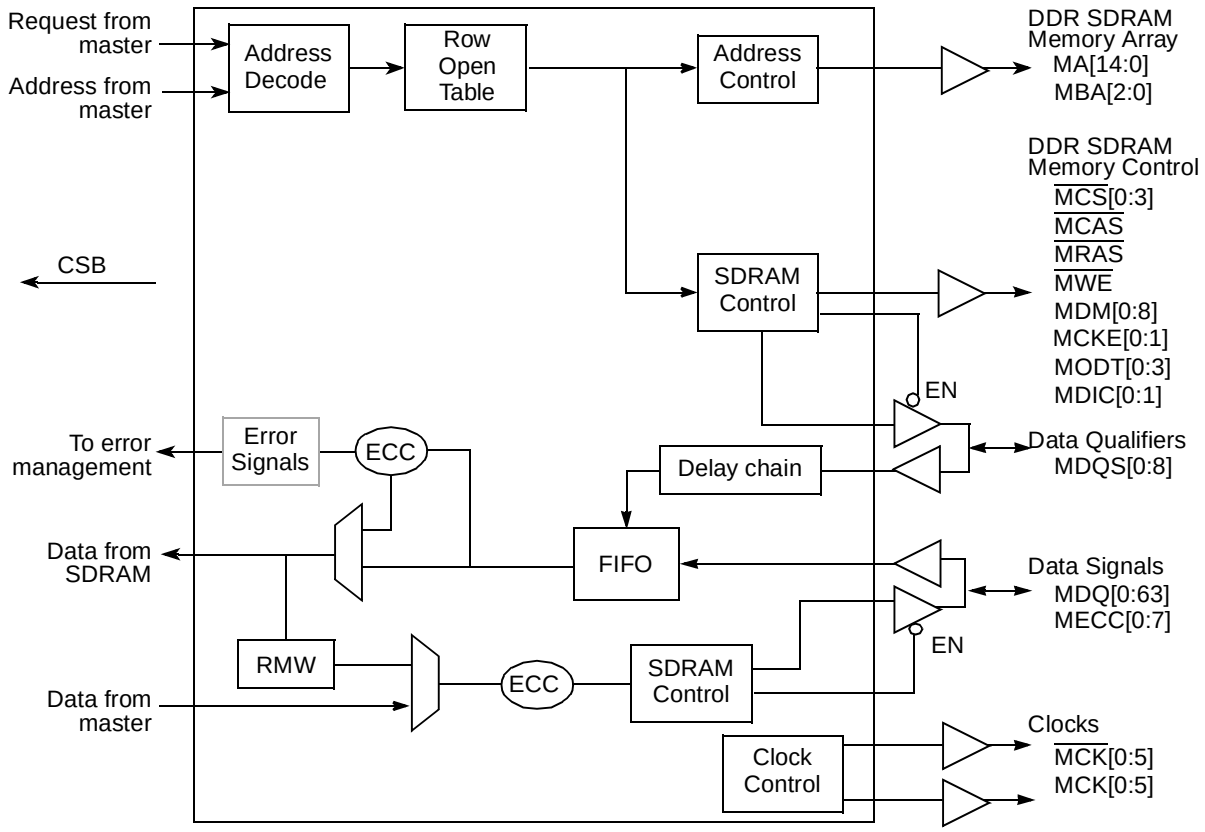


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR SDRAM
- Dual independent controllers—primary and secondary. Both controllers support 32-bit memories. If only the primary controller is used, it can support 64-bit memories as well.
- 64-/72-bit SDRAM data bus (when using only the primary controller). 32-/40-bit SDRAM data bus supported (when using both primary and secondary controllers) for DDR and DDR2
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
  - As many as four physical banks (chip selects), each bank independently addressable
    - When the secondary DDR memory controller is enabled, both primary and secondary DDR memory controllers each support as many as two physical banks (chip selects), each bank independently addressable.
    - When the secondary memory controller is disabled, the primary DDR memory controller supports as many as four physical banks (chip selects), each bank independently addressable.

  - 64-Mbit to 4-Gbit devices depending on internal device configuration with x8/x16/x32 data ports (no direct x4 support)
  - Unbuffered and registered DIMMs
- Chip select interleaving support
- Support for data mask signals and read-modify-write for sub-double-word writes. Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

9.2.1 Modes of Operation

The DDR memory controller supports the following modes:

- 64-bit memories are only supported by the primary DDR memory controller and only when the secondary controller is disabled.

  In order to use both controllers, they must be set to 32-bit bus mode. This is done by setting DDR_SDRAM_CFG[32_BE] for both controllers. Note that when using DDR1 in 32-bit mode, burst length should be set to 8 beats. This can be done by setting DDR_SDRAM_CFG[8_BE].

  Also note that if 32-bit mode is selected in the primary DDR memory controller, the higher data bits (MDQ[32:63]) are used for the secondary DDR memory controller. Other than this, all of the board connections and register programming are identical to that when 64-bit bus mode is used.
- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing DDR_SDRAM_INTERVAL[BSTOPRE] causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting CS*n*_CONFIG[AP_*n*_EN].

9.3 External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

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[0:63]	Data bus	All zeros	64	I/O
MDQS[0:8]	Data strobes	All zeros	9	I/O
MECC[0:7]	Error checking and correcting	All zeros	8	I/O
$\overline{\text{MCAS}}$	Column address strobe	One	1	O
MA[14:0]	Address bus	All zeros	15	O
MBA[2:0]	Logical bank address	All zeros	3	O
$\overline{\text{MCS}}$[0:3]	Chip selects	All ones	4	O
$\overline{\text{MWE}}$	Write enable	One	1	O
$\overline{\text{MRAS}}$	Row address strobe	One	1	O
MDM[0:8]	Data mask	All zeros	9	O
MCK[0:5]	DRAM clock outputs	All zeros	6	O
$\overline{\text{MCK}}$[0:5]	DRAM clock outputs (complement)	All zeros	6	O
MCKE[0:1]	DRAM clock enable	All zeros	2	O
MODT[0:3]	DRAM on-die termination	All zeros	4	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	O
MDIC[0:1]	Driver impedance calibration	High Z	2	I/O

Table 9-2 shows the memory address signal mappings.

Table 9-2. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR)[1]
	MA9	A9
	MA8	A8 (alternate AP for DDR)[2]
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

1 Auto-precharge for DDR signaled on A10 when DDR_SDRAM_CFG[PCHB8] = 0.

2 Auto-precharge for DDR signaled on A8 when DDR_SDRAM_CFG[PCHB8] = 1.

9.3.2 Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate a s described below.	
		State Meaning	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation—The DDR SDRAM drives data during a READ transaction. High impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.
MDQS[0:8]	I/O	Data strobes. Inputs with read data, outputs with write data.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		State Meaning	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 9-36 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 9-36 for byte lane assignments.
		Timing	Assertion/Negation—If a READ command is registered at clock edge *n*, and the latency is programmed in TIMING_CFG_1[CASLAT] to be *m* clocks, data strobes at the DRAM assert coincident with the data on clock edge *n + m*. See the JEDEC DDR SDRAM specification for more information.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller's bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See Section 9.5.11, "Error Checking and Correcting (ECC)," and Section 23.4.2.2, "Debug Information on ECC Pins," for more details.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR controller on writes.
		Timing	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		State Meaning	Asserted/Negated—Represents the state of ECC being driven by the DDR SDRAMs on reads.
		Timing	Assertion/Negation—Same timing as MDQ High impedance—Same timing as MDQ
MA[14:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[14:0] carry 15 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 9-39 and Table 9-87 for 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 $\overline{\text{MCS}}n$ is active). High impedance—When the memory controller is disabled
MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		State Meaning	Asserted/Negated—Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 9-39 and Table 9-87 describe the mapping of these signals in all cases.
		Timing	Assertion/Negation—Same timing as MA*n* High impedance—Same timing as MA*n*

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{MCAS}}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{\text{MCAS}}$ is asserted for read or write transactions and for mode register set, refresh, and precharge commands.	
		State Meaning	Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 9-44 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 9-44 for more information on the states required on $\overline{\text{MRAS}}$ for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		Timing	Assertion/Negation—Timing is directed by the values described in Section 9.4.1.4, "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.
$\overline{\text{MCS}}$[0:3]	O	Chip selects. Four chip selects supported by the memory controller.	
		State Meaning	Asserted—Selects a physical SDRAM bank to perform a memory operation as described in Section 9.4.1.1, "Chip Select Memory Bounds (CSn_BNDS)," and Section 9.4.1.2, "Chip Select Configuration (CSn_CONFIG)." The DDR controller asserts one of the $\overline{\text{MCS}}$[0:3] signals to begin a memory cycle. Negated—Indicates no SDRAM action during the current cycle.
		Timing	Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_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 9-44 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.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB) and MDM7 corresponds to the LSB, while MDM8 corresponds to the ECC byte. Table 9-36 shows byte lane encodings.	
		State Meaning	Asserted—Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDM*n* signals are active-high for the DDR controller. MDM*n* is part of the DDR command encoding. Negated—Allows the corresponding byte to be read from or written to the SDRAM.
		Timing	Assertion/Negation—Same timing as MDQx as outputs. High impedance—Always driven unless the memory controller is disabled.
MODT[0:3]	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT[0:3] represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		State Meaning	Asserted/Negated—Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS*n*_CONFIG[ODT_RD_CFG] and CS*n*_CONFIG[ODT_WR_CFG] fields. High impedance—Always driven.
MDIC[0:1]	I/O	Driver impedance calibration. Note that the MDIC signals require the use of 18.2-Ω precision 1% resistors; MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV_{DD}. SeeSection 5.4.3.8, “DDR Control Driver Register (DDRCDR),” for more information on these signals.	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These will be driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.

9.3.2.2 Clock Interface Signals

Table 9-4 contains the detailed descriptions of the clock signals of the DDR controller.

Table 9-4. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:5], $\overline{\text{MCK}}$[0:5]	O	DRAM clock outputs and their complements. See Section 9.5.4.1, “Clock Distribution.”Enabled by the MCKENR 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.
MCKE[0:1]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0:1] can be negated to stop clocking the DDR SDRAM.	
		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[0:1] are negated.
		Timing	Assertion/Negation—Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Will be 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 inSection 5.3.2.7, “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 Registers—Block Base Address: 0x0_2000 for primary controller; 0x0_D000 for secondary controller				
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	9.4.1.1/9-11
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	9.4.1.2/9-12
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.3/9-14
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-15
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-17
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-19
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.7/9-21
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.8/9-23
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.9/9-25
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.10/9-26
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.11/9-26
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.12/9-29
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.13/9-30
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.14/9-30

Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	9.4.1.15/9-31
0x150–0xBF4	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	9.4.1.16/9-31
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	9.4.1.17/9-32
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.18/9-32
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.19/9-33
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.20/9-33
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.21/9-34
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.22/9-34
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.23/9-35
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.24/9-35
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.25/9-36
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.26/9-37
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.27/9-38
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.28/9-39
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.29/9-39

[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 (CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS*n*_BNDS should equal the size of physical DRAM. Also, note that EA*n* must be greater than or equal to SA*n*.

If chip select interleaving is enabled, all fields in the lower interleaved chip select will be used, and the other chip selects' bounds registers will be unused. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS will be used, and all fields in CS1_BNDS will be unused.

CS*n*_BNDS are shown in Figure 9-2.

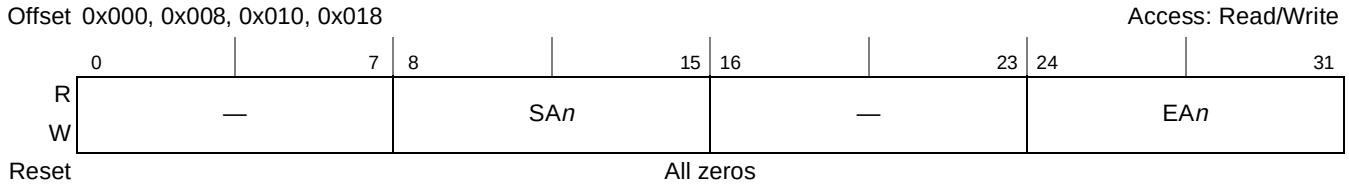


Figure 9-2. Chip Select Bounds Registers (CS*n*_BNDS)

Table 9-6 describes the CS*n*_BNDS register fields.

Table 9-6. CS*n*_BNDS Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SA*n*	Starting address for chip select (bank) *n*. This value is compared against the 8 msbs of the 32-bit address.
16–23	—	Reserved
24–31	EA*n*	Ending address for chip select (bank) *n*. This value is compared against the 8 msbs of the 32-bit address.

9.4.1.2 Chip Select Configuration (CS*n*_CONFIG)

The chip select configuration (CS*n*_CONFIG) registers shown in Figure 9-3 enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because CS*n*_CONFIG[ROW_BITS_CS_*n*, COL_BITS_CS_*n*] establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused, with the exception of the ODT_RD_CFG and ODT_WR_CFG fields. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_CONFIG will be used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG will be used.

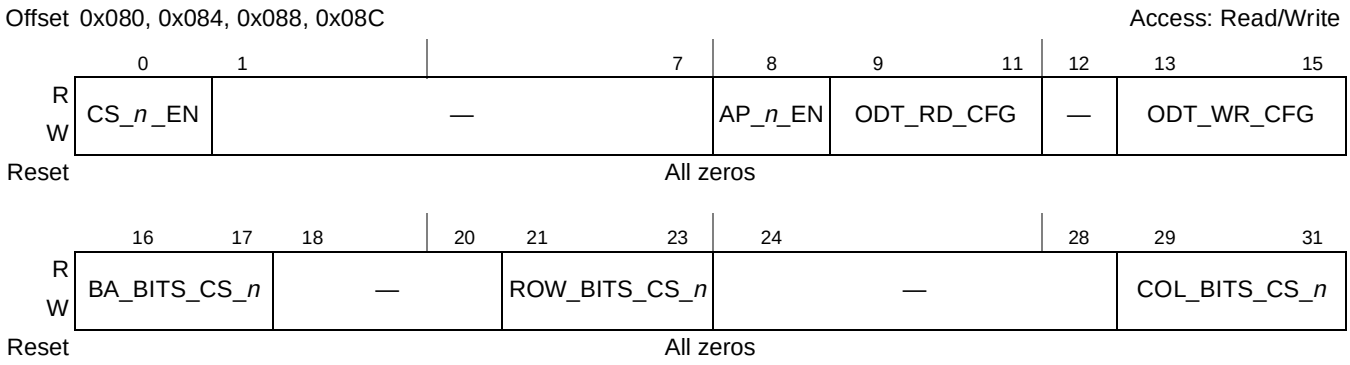


Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

Table 9-7 describes the CS*n*_CONFIG register fields.

Table 9-7. CS*n*_CONFIG Field Descriptions

Bits	Name	Description
0	CS_*n*_EN	Chip select *n* enable 0 Chip select *n* is not active 1 Chip select *n* is active and assumes the state set in CS*n*_BNDS.
1–7	—	Reserved
8	AP_*n*_EN	Chip select *n* auto-precharge enable 0 Chip select *n* will only be auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n* will always issue an auto-precharge for read and write transactions.
9–11	ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS*n* 010 Assert ODT only during reads to other chip selects 011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads 101–111Reserved
12	—	Reserved
13–15	ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT _WR_CFG to be enabled. ODT should only be used with DDR2 memories. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS*n* 010 Assert ODT only during writes to other chip selects 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes 101–111Reserved
16–17	BA_BITS_CS_*n*	Number of bank bits for SDRAM on chip select *n*. These bits correspond to the sub-bank bits driven on MBA*n* in Table 9-38Table 9-39and Table 9-40. 00 2 logical bank bits 01 3 logical bank bits 10–11Reserved
18–20	—	Reserved
21–23	ROW_BITS_CS_*n*	Number of row bits for SDRAM on chip select *n*. See Table 9-38Table 9-39and Table 9-40 for details. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 000–111 Reserved

Table 9-7. CS*n*_CONFIG Field Descriptions (continued)

Bits	Name	Description
24–28	—	Reserved
29–31	COL_BITS_CS_*n*	Number of column bits for SDRAM on chip select *n*. For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100–111 Reserved

9.4.1.3 DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

DDR SDRAM timing configuration register 3, shown in Figure 9-4, sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

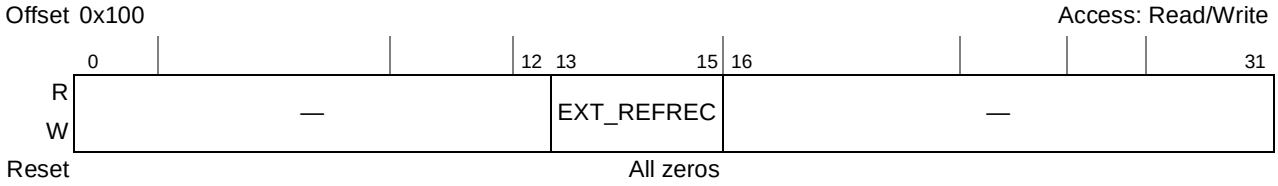


Figure 9-4. DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

Table 9-8 describes TIMING_CFG_3 fields.

Table 9-8. TIMING_CFG_3 Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13–15	EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain a 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7-bit value of the refresh recovery. t_{RFC} = {EXT_REFREC || 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 1	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 will be added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller will use 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 will determine 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 10 2 clocks 01 1 clock 11 3 clocks
2–3	WRT	Write-to-read turnaround. Specifies how many extra cycles will be added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller will use 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 will determine 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 10 2 clocks 01 1 clock 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles will be added between reads to different chip selects. As a default, 3 cycles will be 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 will be the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles will be added between writes to different chip selects. As a default, 2 cycles will be 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 will be the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 10 2 clocks 01 1 clock 11 3 clocks
8	—	Reserved, should be cleared.

Table 9-9. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
9–11	ACT_PD_EXIT	Active powerdown exit timing (t_{XARD} and t_{XARDS}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 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.
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 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks

9.4.1.5 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 9-6, sets the number of clock cycles between various SDRAM control commands.

Offset 0x108 Access: Read/Write

	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.
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

Table 9-10. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
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
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.

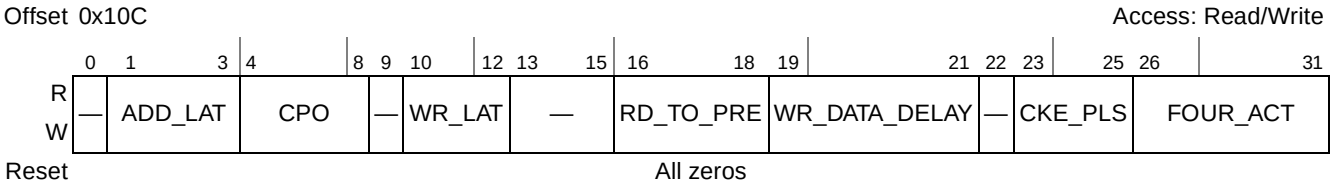


Figure 9-7. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

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. 00000READ_LAT + 1 00001Reserved 00010READ_LAT 00011READ_LAT + 1/4 00100READ_LAT + 1/2 00101READ_LAT + 3/4 00110READ_LAT + 1 00111READ_LAT + 5/4 01000READ_LAT + 3/2 01001READ_LAT + 7/4 01010READ_LAT + 2 01011READ_LAT + 9/4 01100READ_LAT + 5/2 01101READ_LAT + 11/4 01110READ_LAT + 3 01111READ_LAT + 13/4 10000READ_LAT + 7/2 10001READ_LAT + 15/4 10010READ_LAT + 4 10011READ_LAT + 17/4 10100READ_LAT + 9/2 10101READ_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

Table 9-11. TIMING_CFG_2 Field Descriptions (continued)

Bits	Name	Description
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
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. 000000Reserved 0000011 cycle 0000102 cycles 0000113 cycles 0001004 cycles ... 01001119 cycles 01010020 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.

Offset 0x110 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MEM_EN	SREN	ECC_EN	RD_EN	—	SDRAM_TYPE			—		DYN_PWR	—	32_BE	8_BE	NCAP	—
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	2T_EN	BA_INTLV_CTL							—		x32_EN	PCHB8	HSE	—	MEM_HALT	BI
Reset	All zeros															

Figure 9-8. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 9-12 describes the DDR_SDRAM_CFG fields.

Table 9-12. DDR_SDRAM_CFG Field Descriptions

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.
2	ECC_EN	ECC enable. Note that uncorrectable read errors may cause an interrupt. 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs. Note that 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 will be used when issuing the automatic hardware initialization sequence to DRAM via 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

Table 9-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
8–9	—	Reserved
10	DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11	—	Reserved
12	32_BE	32-bit bus enable. 0 64-bit bus is used. 1 32-bit bus is used.
13	8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. **Note:** 8-beat bursts may be used by DDR1 (SDRAM_TYPE = 010), and only when using 32-bit bus mode (32_BE = 1); 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 will need to be set if auto precharge is used. 0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
15	—	Reserved
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address 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 that RD_EN and 2T_EN must not both be set at the same time.
17–23	BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('*x*' denotes a don't care bit value. All unlisted field values are reserved.) 0000000No external memory banks are interleaved 1000000External memory banks 0 and 1 are interleaved 0100000External memory banks 2 and 3 are interleaved 1100000External memory banks 0 and 1 are interleaved together **and** banks 2 and 3 are interleaved together *xx*00100External memory banks 0 through 3 are all interleaved together
24–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 will be used to capture DQ[0:31], DQS4 will be used to capture DQ[32:63] and DQS8 will be used to capture ECC[0:7].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] will be used to indicate the auto-precharge and precharge all commands. 1 MA[8] will be used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.

Table 9-12. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance will be used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in Section 5.3.2.8, "DDR Control Driver Register (DDRCDR)." 0 I/O driver impedance will be configured to full strength. 1 I/O driver impedance will be configured to half strength.
29	—	Reserved
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller will 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 will accept new transactions. 1 DDR controller will finish any remaining transactions, and then it will remain halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller will cycle through initialization routine based on SDRAM_TYPE 1 Initialization routine will be 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 will not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled. See Section 9.4.1.15, "DDR Initialization Address (DDR_INIT_ADDR)," for details on avoiding ECC errors in this mode.

9.4.1.8 DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2, shown in Figure 9-9, provides more control configuration for the DDR controller.

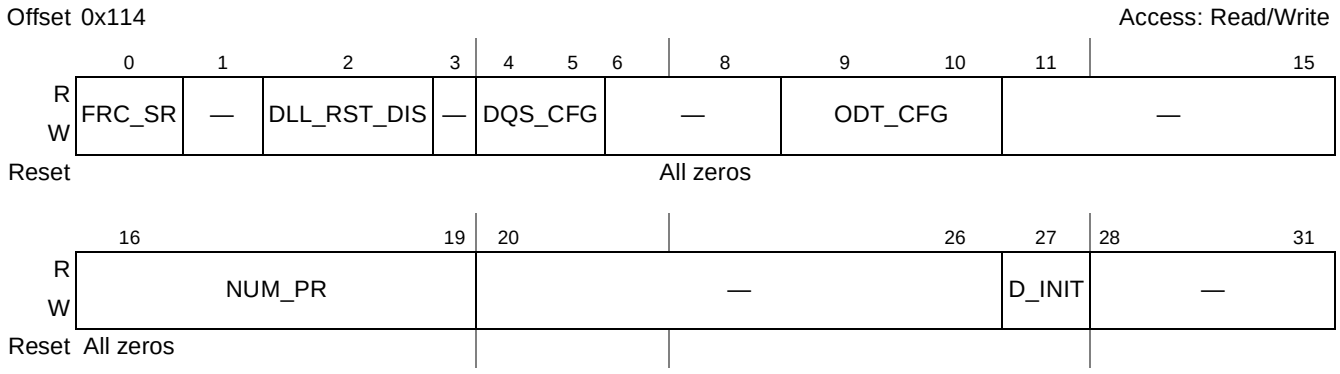


Figure 9-9. DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)

Table 9-13 describes the DDR_SDRAM_CFG_2 fields.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions

Bits	Name	Description
0	FRC_SR	Force self refresh 0 DDR controller will operate in normal mode. 1 DDR controller will enter self-refresh mode.
1	—	Reserved. Should be cleared.
2	DLL_RST_DIS	DLL reset disable. The DDR controller will typically issue 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 will issue a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller will not issue a DLL reset to the DRAMs when exiting self refresh.
3	—	Reserved
4–5	DQS_CFG	DQS configuration 00 Only true DQS signals are used. 01 Reserved 10 Reserved 11 Reserved
6–8	—	Reserved
9–10	ODT_CFG	ODT configuration. This field defines how ODT will be driven to the on-chip IOs. SeeSection 5.3.2.8, “DDR Control Driver Register (DDRCDR),” which defines the termination value that will be used. (DDR2-specific, must be cleared for DDR1) 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15	—	Reserved
16–19	NUM_PR	Number of posted refreshes. This will determine 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 will not be 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 will be issued at a time 0010 2 refreshes will be issued at a time 0011 3 refreshes will be issued at a time ... 1000 8 refreshes will be issued at a time 1001–1111Reserved
20–26	—	Reserved, should be cleared.

Table 9-13. DDR_SDRAM_CFG_2 Field Descriptions (continued)

Bits	Name	Description
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 will automatically initialize DRAM after it is enabled. This bit will be automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller will initialize memory once it is enabled. This bit will remain asserted until the initialization is complete. The value in DDR_DATA_INIT register will be 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.

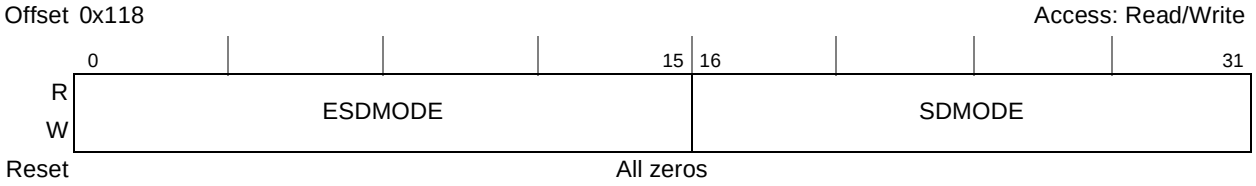


Figure 9-10. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

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, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

9.4.1.10 DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register, shown in Figure 9-11, sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2).

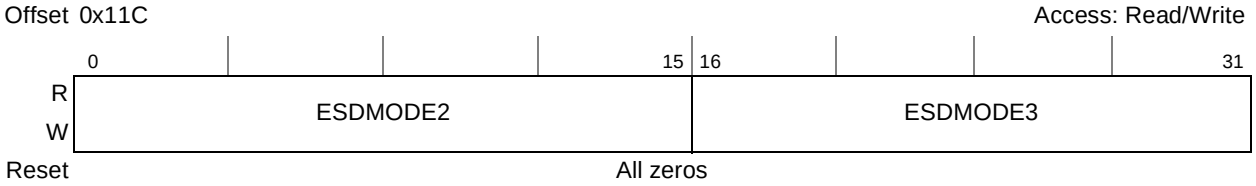


Figure 9-11. DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)

Table 9-15 describes the DDR_SDRAM_MODE_2 fields.

Table 9-15. DDR_SDRAM_MODE_2 Field Descriptions

Bits	Name	Description
0–15	ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31	ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in Figure 9-11, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

9.4.1.11 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register, shown in Figure 9-12, allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signals to a specific value

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.

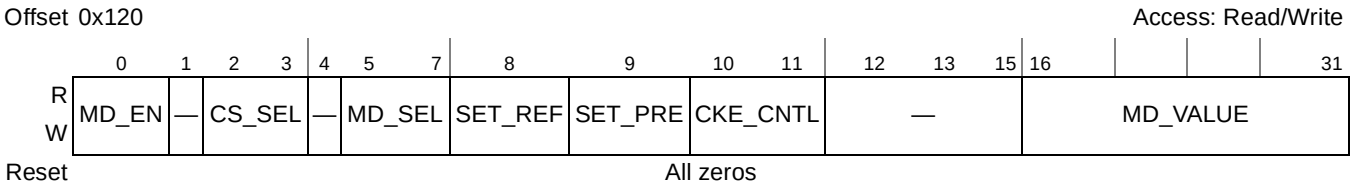


Figure 9-12. DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

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. In addition, the chip select must be chosen by setting CS_SEL. MD_EN is set by software and cleared by hardware once the command has been issued. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1	—	Reserved
2–3	CS_SEL	Select chip select. Specifies the chip select that will be driven active due to any command forced by software in DDR_SDRAM_MD_CNTL. 00 Chip select 0 is active 01 Chip select 1 is active 10 Chip select 2 is active 11 Chip select 3 is active
4	—	Reserved
5–7	MD_SEL	Mode register select. MD_SEL specifies one of the following: • 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 will be presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3
8	SET_REF	Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be 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 to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be 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.

Table 9-16. DDR_SDRAM_MD_CNTL Field Descriptions (continued)

Bits	Name	Description
10–11	CKE_CNTL	Clock enable control. Allows software to globally clear or set all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value will continue to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller will continue to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (i.e., 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 will be presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

Table 9-17 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 9-17. Settings of DDR_SDRAM_MD_CNTL Fields

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	—
SET_REF	0	1	0	—
SET_PRE	0	0	1	—
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See Table 9-16.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 9-16.	—
CKE_CNTL	0	0	0	See Table 9-16.

9.4.1.12 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 9-13, sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.

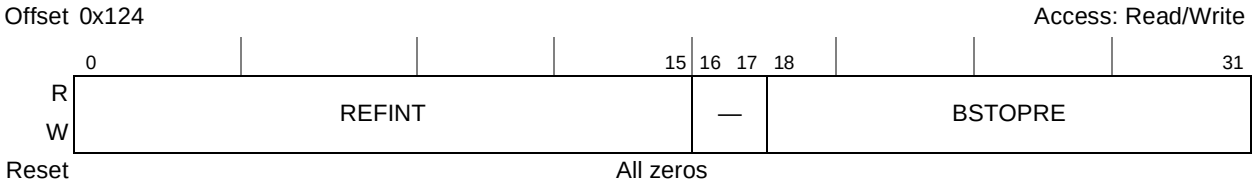


Figure 9-13. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 9-18 describes the DDR_SDRAM_INTERVAL fields.

Table 9-18. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes will not be 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 will be used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

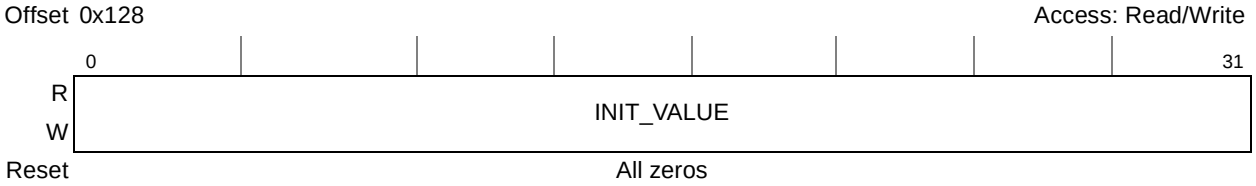


Figure 9-14. DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)

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 will be 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.

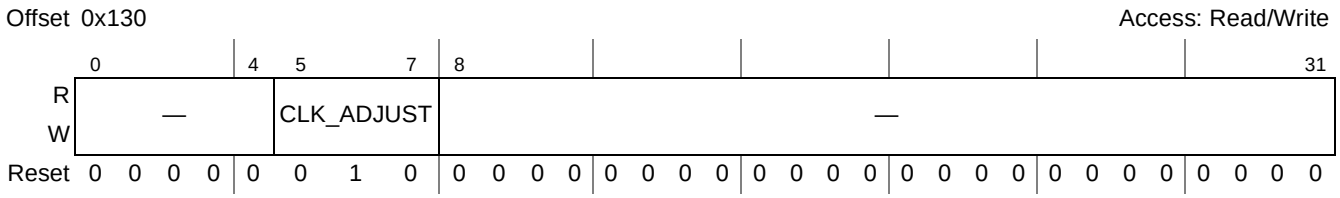


Figure 9-15. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

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 will be launched aligned with address/command 001 Clock will be launched 1/4 applied cycle after address/command 010 Clock will be launched 1/2 applied cycle after address/command 011 Clock will be launched 3/4 applied cycle after address/command 100 Clock will be launched 1 applied cycle after address/command 101–111Reserved
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 will be used for the automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

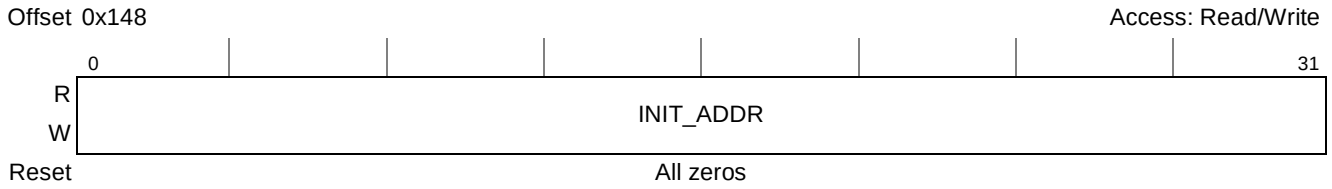


Figure 9-16. DDR Initialization Address Configuration Register (DDR_INIT_ADDR)

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 will be used for the automatic CAS to preamble calibration at POR.

9.4.1.16 DDR IP Block Revision 1 (DDR_IP_REV1)

The DDR IP block revision 1 register, shown in Figure 9-17, provides read-only fields with the IP block ID, along with major and minor revision information.

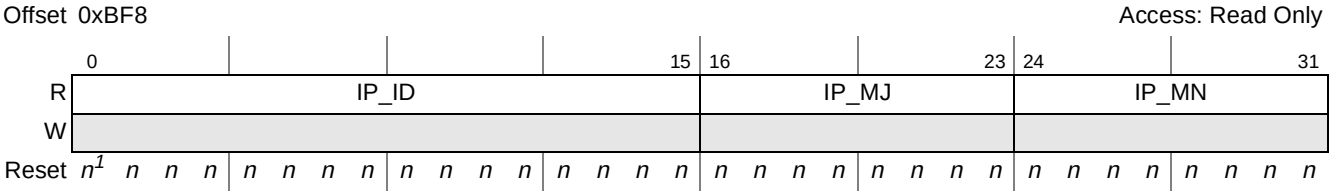


Figure 9-17. DDR IP Block Revision 1 (DDR_IP_REV1)

1 For reset values, see Table 9-22.

Table 9-22 describes the DDR_IP_REV1 fields.

Table 9-22. DDR_IP_REV1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID. For the DDR controller, this value is 0x0002.

9.4.1.17 DDR IP Block Revision 2 (DDR_IP_REV2)

The DDR IP block revision 2 register, shown in Figure 9-18, provides read-only fields with the IP block integration and configuration options.

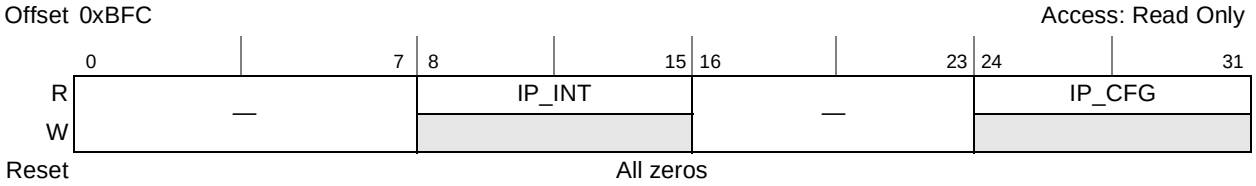


Figure 9-18. DDR IP Block Revision 2 (DDR_IP_REV2)

Table 9-23 describes the DDR_IP_REV2 fields.

Table 9-23. DDR_IP_REV2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

9.4.1.18 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in Figure 9-19.

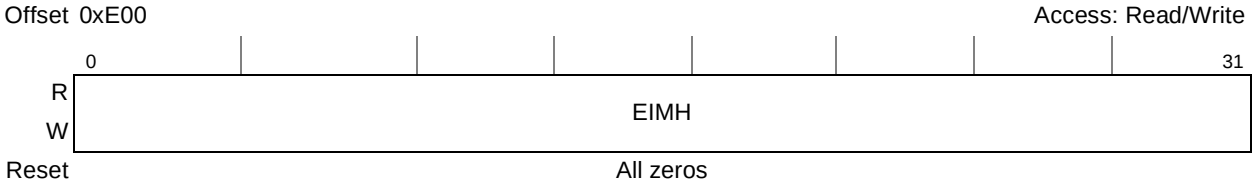


Figure 9-19. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

Table 9-24 describes the DATA_ERR_INJECT_HI fields.

Table 9-24. DATA_ERR_INJECT_HI Field Descriptions

Bits	Name	Description
0–31	EIMH	Error injection mask high data path. Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.19 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in Figure 9-20.

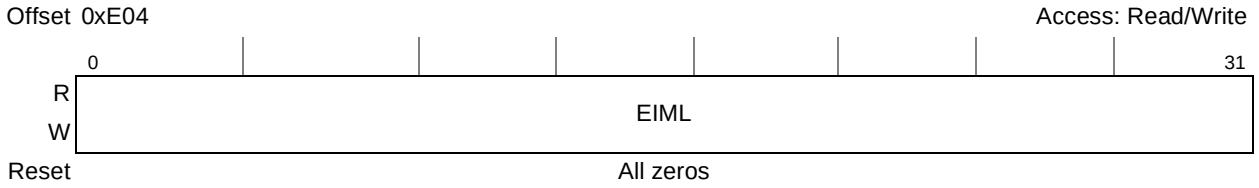


Figure 9-20. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

Table 9-25 describes the DATA_ERR_INJECT_LO fields.

Table 9-25. DATA_ERR_INJECT_LO Field Descriptions

Bits	Name	Description
0–31	EIML	Error injection mask low data path. Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.20 Memory Data Path Error Injection Mask ECC (ERR_INJECT)

The memory data path error injection mask ECC register, shown in Figure 9-21, sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.

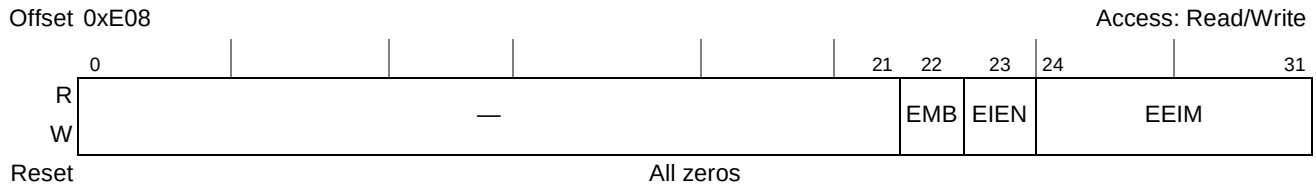


Figure 9-21. Memory Data Path Error Injection Mask ECC Register (ERR_INJECT)

Table 9-26 describes the ERR_INJECT fields.

Table 9-26. ERR_INJECT Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	EMB	ECC mirror byte 0 Mirror byte functionality disabled. 1 Mirror the most significant data path byte onto the ECC byte.
23	EIEN	Error injection enable 0 Error injection disabled. 1 Error injection enabled. This applies to the data mask bits, the ECC mask bits, and the ECC mirror bit. Note that error injection should not be enabled until the memory controller has been enabled via DDR_SDRAM_CFG[MEM_EN].
24–31	EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

9.4.1.21 Memory Data Path Read Capture High (CAPTURE_DATA_HI)

The memory data path read capture high register, shown in Figure 9-22, stores the high word of the read data path during error capture.

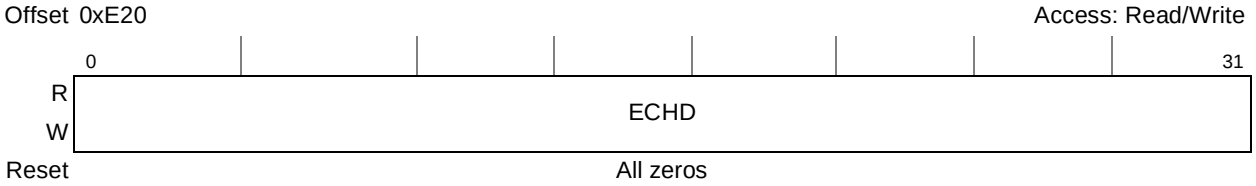


Figure 9-22. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 9-27 describes the CAPTURE_DATA_HI fields.

Table 9-27. CAPTURE_DATA_HI Field Descriptions

Bits	Name	Description
0–31	ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.

9.4.1.22 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in Figure 9-23, stores the low word of the read data path during error capture.

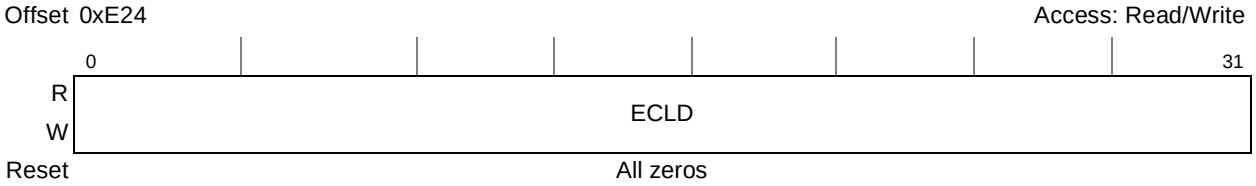


Figure 9-23. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 9-28 describes the CAPTURE_DATA_LO fields.

Table 9-28. CAPTURE_DATA_LO Field Descriptions

Bits	Name	Description
0–31	ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

9.4.1.23 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in Figure 9-24, stores the ECC syndrome bits that were on the data bus when an error was detected.

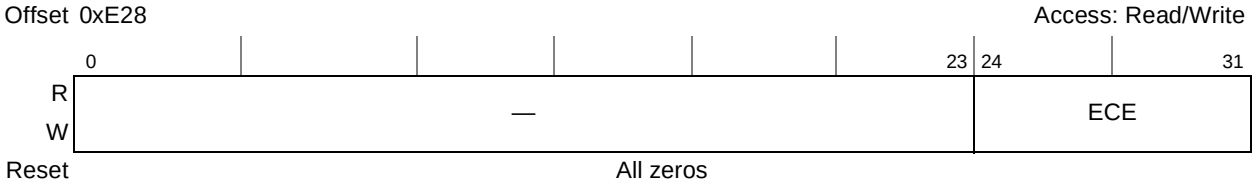


Figure 9-24. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-29 describes the CAPTURE_ECC fields.

Table 9-29. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected.

9.4.1.24 Memory Error Detect (ERR_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

ERR_DETECT is shown in Figure 9-25.

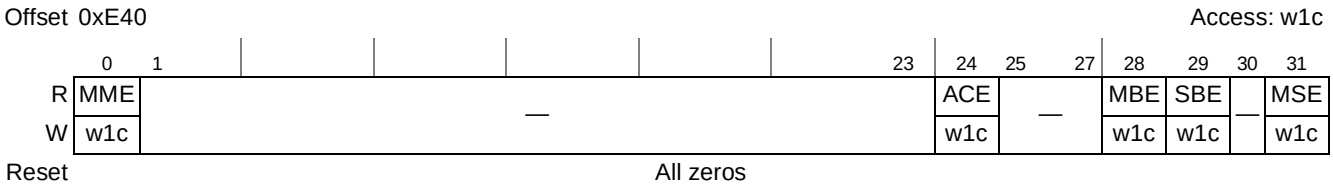


Figure 9-25. Memory Error Detect Register (ERR_DETECT)

Table 9-30 describes the ERR_DETECT fields.

Table 9-30. ERR_DETECT Field Descriptions

Bits	Name	Description
0	MME	Multiple memory errors. This bit is cleared by software writing a 1. 0 Multiple memory errors of the same type were not detected. 1 Multiple memory errors of the same type were detected.
1–23	—	Reserved

Table 9-30. ERR_DETECT Field Descriptions (continued)

Bits	Name	Description
24	ACE	Automatic calibration error. This bit is cleared by software writing a 1. 0 An automatic calibration error has not been detected. 1 An automatic calibration error has been detected.
25–27	—	Reserved
28	MBE	Multiple-bit error. This bit is cleared by software writing a 1. 0 A multiple-bit error has not been detected. 1 A multiple-bit error has been detected.
29	SBE	Single-bit ECC error. This bit is cleared by software writing a 1. 0 The number of single-bit ECC errors detected has not crossed the threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30	—	Reserved
31	MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

9.4.1.25 Memory Error Disable (ERR_DISABLE)

The memory error disable register, shown in Figure 9-26, allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

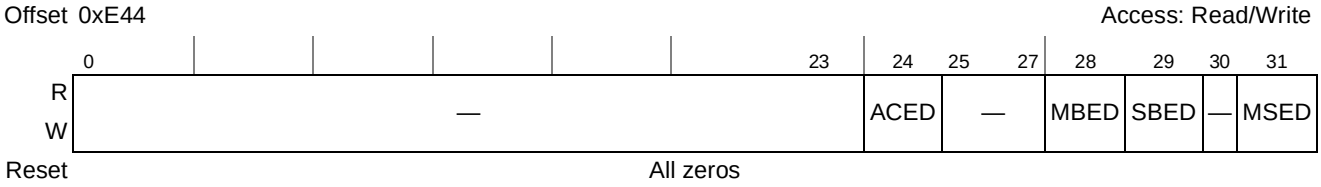


Figure 9-26. Memory Error Disable Register (ERR_DISABLE)

Table 9-31 describes the ERR_DISABLE fields.

Table 9-31. ERR_DISABLE Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	ACED	Automatic calibration error disable 0 Automatic calibration errors are enabled. 1 Automatic calibration errors are disabled.
25–27	—	Reserved
28	MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. See Section 9.5.12, "Error Management," for more information. 1 Multiple-bit ECC errors are not detected or reported.
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.

Table 9-31. ERR_DISABLE Field Descriptions (continued)

Bits	Name	Description
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

9.4.1.26 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 9-27, enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

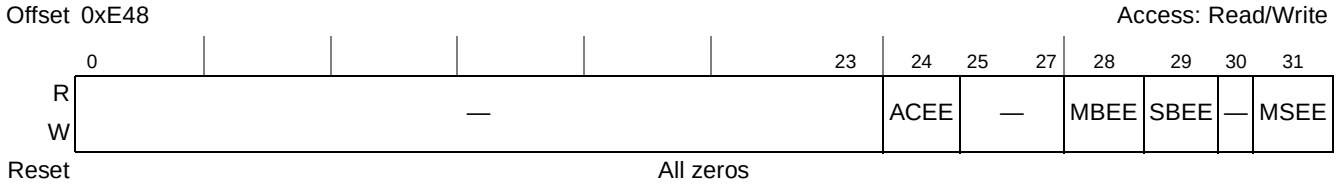


Figure 9-27. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 9-32 describes the ERR_INT_EN fields.

Table 9-32. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	ACEE	Automatic calibration error interrupt enable 0 Automatic calibration errors cannot generate interrupts. 1 Automatic calibration errors generate interrupts.
25–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. See Section 9.5.12, “Error Management,” for more information. 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29	SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

9.4.1.27 Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)

The memory error attributes capture register, shown in Figure 9-28, sets attributes for errors including type, size, source, and others.

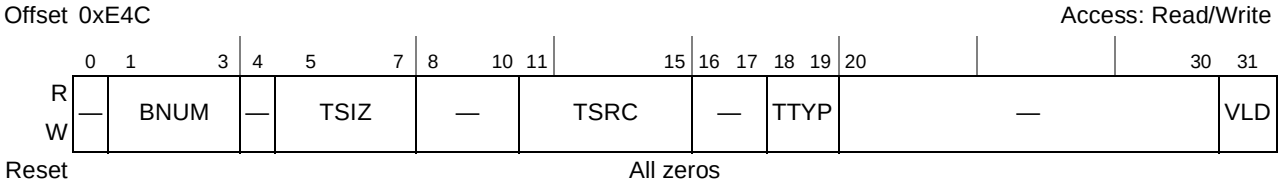


Figure 9-28. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 9-33 describes the CAPTURE_ATTRIBUTES fields.

Table 9-33. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4	—	Reserved
5–7	TSIZ	Transaction size for the error. Captures the transaction size in double words.
8–10	—	Reserved
11–15	TSRC	Transaction source for the error 00000 e300 core data transaction 00001 Reserved 00010 e300 core instruction fetch 00011 Reserved 00100 TSEC 1 00101 TSEC 2 00110 USB MPH 00111 USB DR 01000 Encryption core 01001 I^2C (boot sequencer) 01010 JTAG 01011 Reserved 01100 Reserved 01101 PCI 1 01110 PCI 2 01111 DMA 10000–11111 Reserved
16–17	—	Reserved
18–19	TTYP	Transaction type for the error 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VLD	Valid. Set as soon as valid information is captured in the error capture registers.

9.4.1.28 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 9-29, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

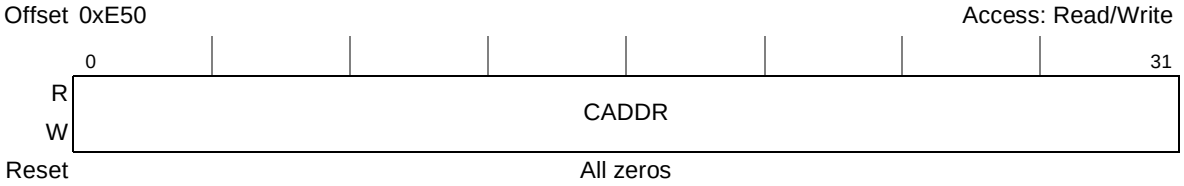


Figure 9-29. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 9-34 describes the CAPTURE_ADDRESS fields.

Table 9-34. CAPTURE_ADDRESS Field Descriptions

Bits	Name	Description
0–31	CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

9.4.1.29 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in Figure 9-30, stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.

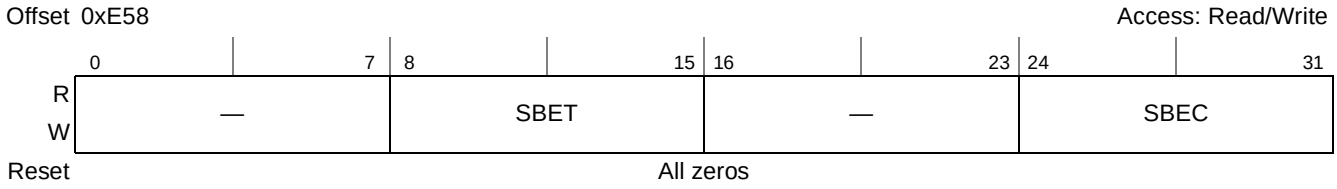


Figure 9-30. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 9-35 describes the ERR_SBE fields.

Table 9-35. ERR_SBE Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.
16–23	—	Reserved
24–31	SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

Figure 9-31 is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40bit wide memory. Bank sizes up to 2 Gbytes (maximum total physical memory size of 4 Gbytes) are supported, providing up to a maximum of 4 Gbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32-bit data bus, detects all double-bit errors within the 64- or 32-bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

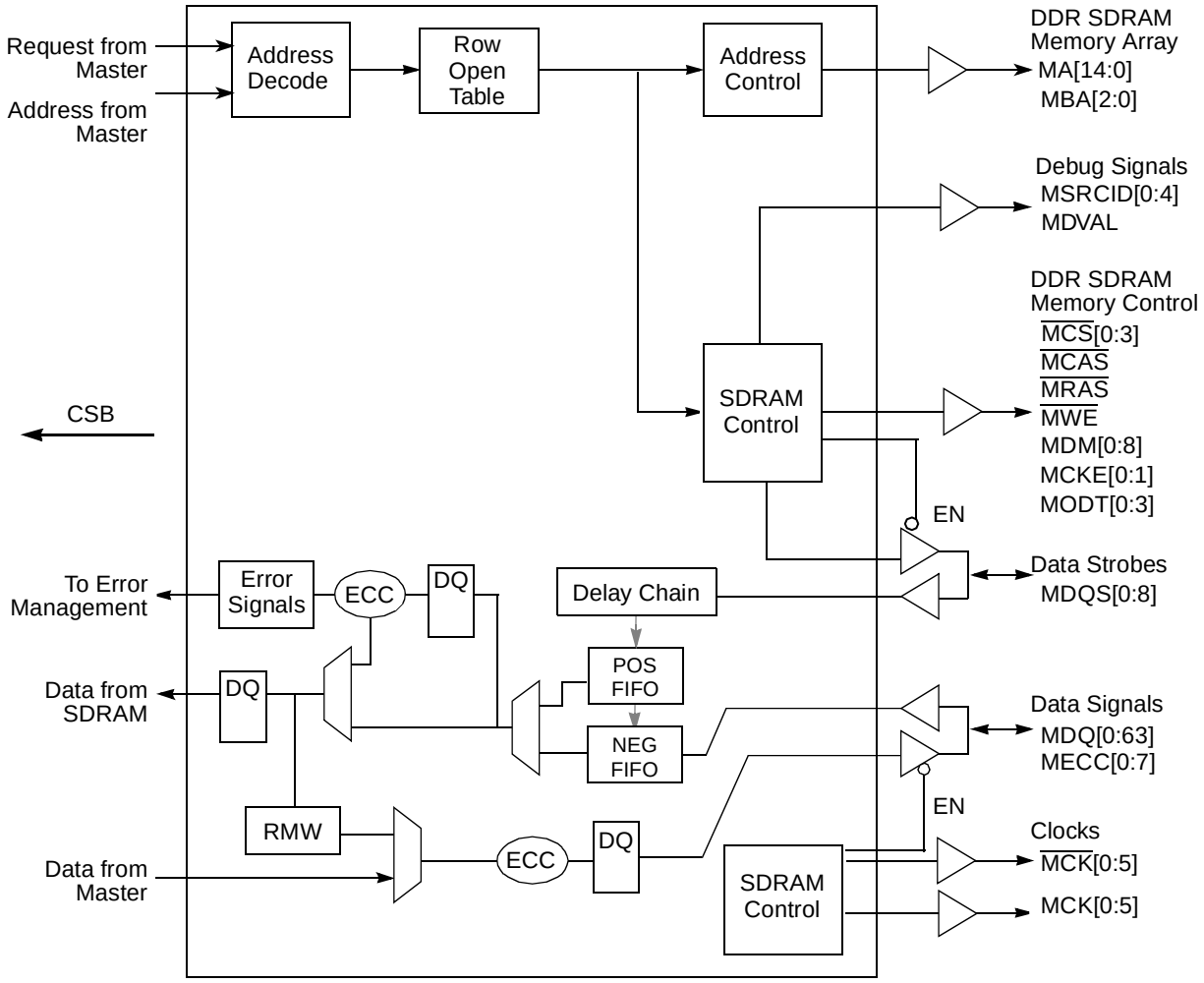


Figure 9-31. DDR Memory Controller Block Diagram

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

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

Figure 9-32 shows an example DDR SDRAM configuration with four logical banks.

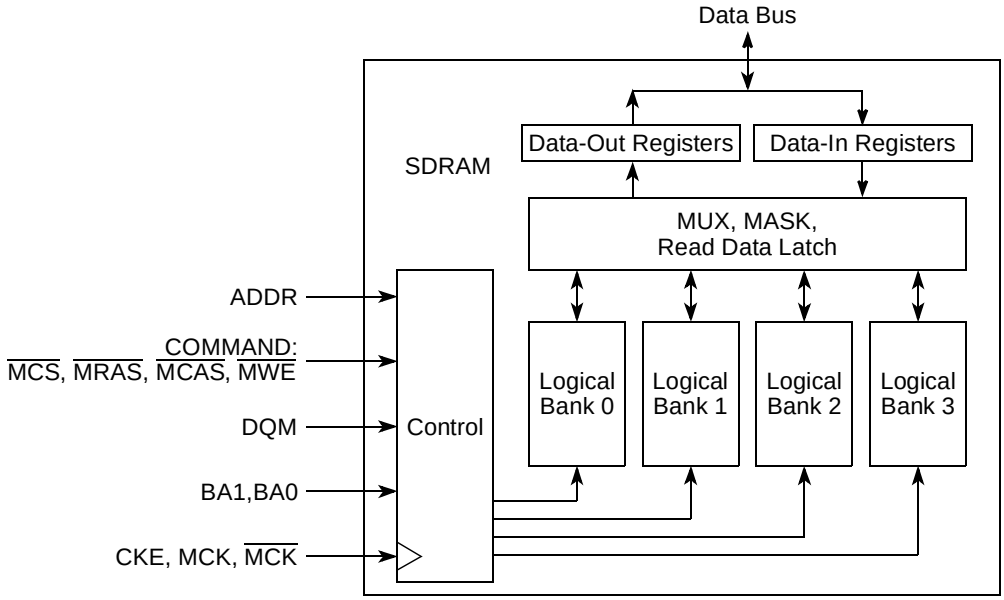


Figure 9-32. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-33 shows some typical signal connections.

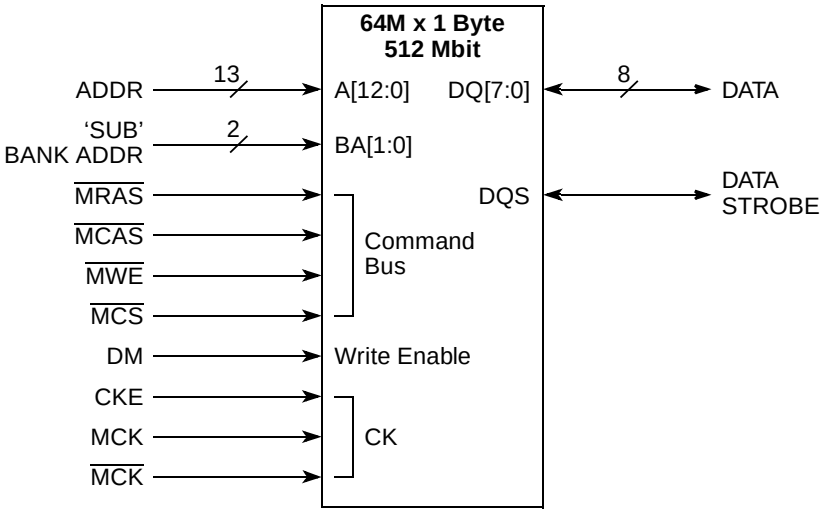
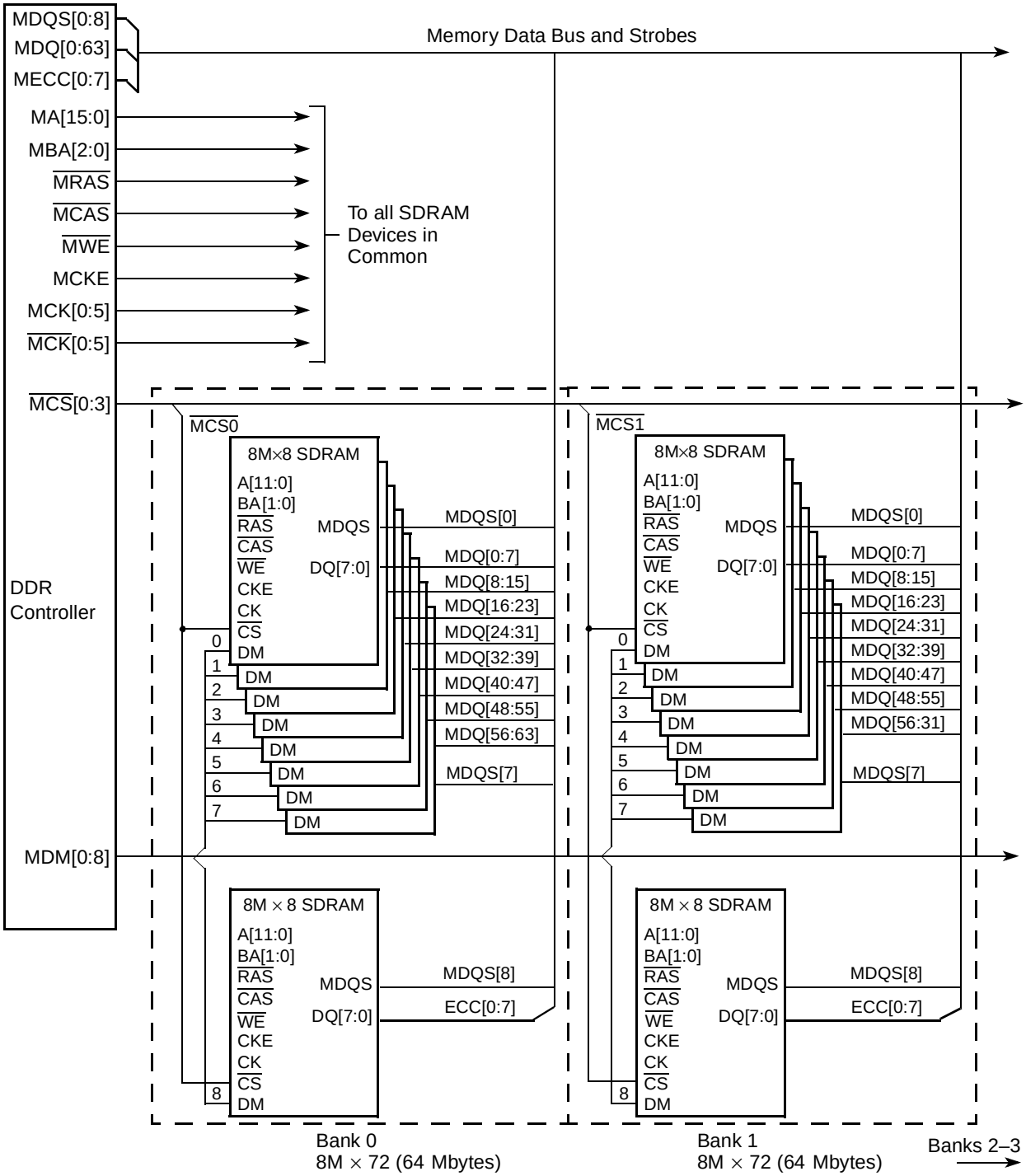


Figure 9-33. Typical DDR SDRAM Interface Signals

Figure 9-34 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M × 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 15 address pins, but in this example the DDR SDRAM devices use only 12 bits.



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

Figure 9-34. Example 256-Mbyte DDR SDRAM Configuration With ECC

Section 9.5.12, "Error Management," explains how the DDR memory controller handles errors.

9.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Fifteen multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 4 Gbits. Four chip select ($\overline{CS}$) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 Mbytes to 4 Gbytes. The physical banks can be constructed using x8, x16, or x32 memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

An 8-bit DDR SDRAM device has a DQM signal and 8 data signals (DQ[0:7]). A 16-bit DDR SDRAM device has 2 DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

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

Table 9-36 shows the DDR memory controller's relationships between data byte lane0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

Table 9-36. Byte Lane to Data Relationship

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

9.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 15 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 15 row address bits, plus 2 or 3 logical bank-select bits and from 8–11 column address bits.

Table 9-37 describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits x 8	12 x 9 x 2	64 Mbytes	256 Mbytes
64 Mbits[1]	4 Mbits x 16	12 x 8 x 2	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbytes
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	13 x 11 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbytes
1 Gbits	128 Mbits x 8	14 x 11 x 2	1 Gbyte	4 Gbytes
1 Gbits	64 Mbits x 16	14 x 10 x 2	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 11 x 2	2 Gbytes	4 Gbytes (two banks)
2 Gbits	128 Mbits x 16	15 x 10 x 2	1 Gbytes	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 11 x 2	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	16 x 10 x 2	2 Gbytes	8 Gbytes

[1] This configuration is not supported in 16-bit bus mode.

Table 9-37. Supported DDR2 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbytes
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbytes
1 Gbits	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	4 Gbytes (two banks)
2 Gbits	128Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	4 Gbytes (two banks)

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in Section 9.5.12, “Error Management.”

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the

DDR memory controller to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}n$ signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

9.5.2 DDR SDRAM Address Multiplexing

Table 9-38, Table 9-39, Table 9-40, and Table 9-41 show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[14:0] use MA[14] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR1/DDR2 modes for reads and writes, so the column address can never use MA[10].

Table 9-38. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col		msb		Address from Core Master																										lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29–31
15 x 11 x 2	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	1	0												
	$\overline{\text{MCAS}}$																			11	9	8	7	6	5	4	3	2	1	0	
15 x 10 x 2	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
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	

Table 9-38. DDR1 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb		Address from Core Master																										lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29–31
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 9-39. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																												lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30–31
15 x 11 x 2	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	$\overline{\text{MCAS}}$																				11	9	8	7	6	5	4	3	2	1	0	
15 x 10 x 2	$\overline{\text{MRAS}}$				14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
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	

Table 9-39. DDR1 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled (continued)

Row x Col		msb	Address from Core Master																												lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30–31
12 x 9 x 2	$\overline{\text{MRAS}}$								11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	
12 x 8 x 2	$\overline{\text{MRAS}}$									11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																					1	0									
	$\overline{\text{MCAS}}$																							7	6	5	4	3	2	1	0	

Table 9-40. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving Disabled

Row x Col		msb		Address from Core Master																											lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29–31
15 x 10 x 3	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	

Table 9-41. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving Disabled

Row x Col		msb	Address from Core Master																												lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30–31
15 x 10 x 3	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
14 x 10 x 3	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
13 x 10 x 3	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		2	1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0
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

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 9-42 illustrates examples of address decode when interleaving between two chip selects, and Table 9-43 shows examples of address decode when interleaving between four chip selects.

Table 9-42. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Two Banks

Row x Col		msb		Address from Core Master																										lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29–31
14 x 10 x 3	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																	2	1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0
14 x 10 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL													
	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	CS SEL														
	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	CS SEL													
	MBA																		1	0										
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0

Table 9-43. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks

Row x Col		msb		Address from Core Master																											lsb
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29–31
14 x 10 x 3	$\overline{\text{MRAS}}$	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	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	CS SEL															
	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	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	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 9-44) 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}, etc.). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR1 memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, $\overline{\text{MCAS}}$ latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory

controller after DDR_SDRAM_CFG[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software via use of the DDR_SDRAM_MD_CNTL register.

- Self refresh (for long periods of standby)—Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller will place all logical banks in a precharged state.

Table 9-44. DDR SDRAM Command Table

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

9.5.4 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in Table 9-45 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-45. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS}.
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD}.
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the read latency is m clocks, the data is available nominally coincident with clock edge $n + m$.
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP}
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as t_{RP}
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 9-35 through Figure 9-37 show DDR SDRAM timing for various types of accesses; see Figure 9-35 for a single-beat read operation, Figure 9-36 for a single-beat write operation, and Figure 9-37 for a 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).

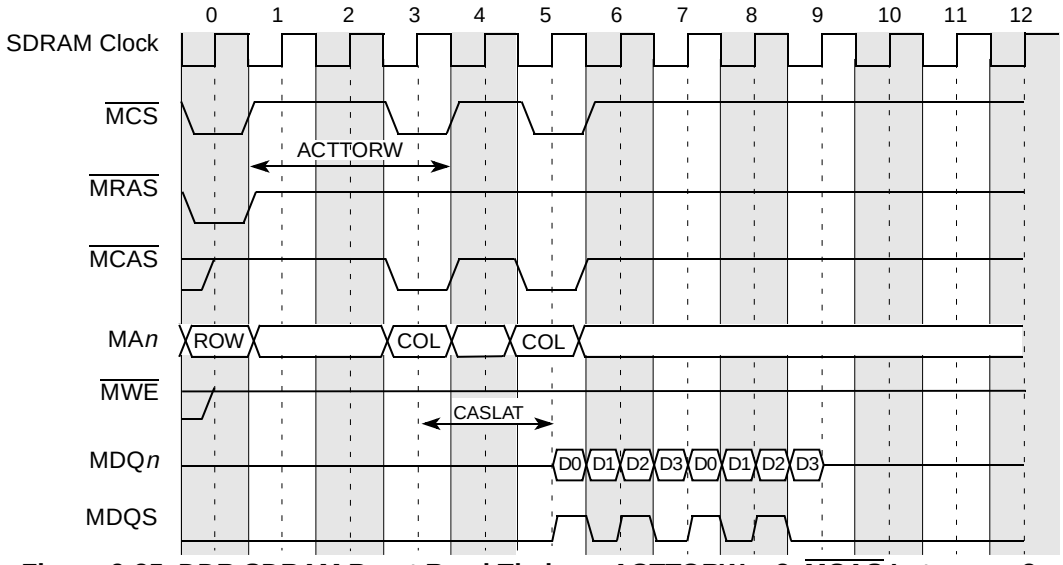


Figure 9-35. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

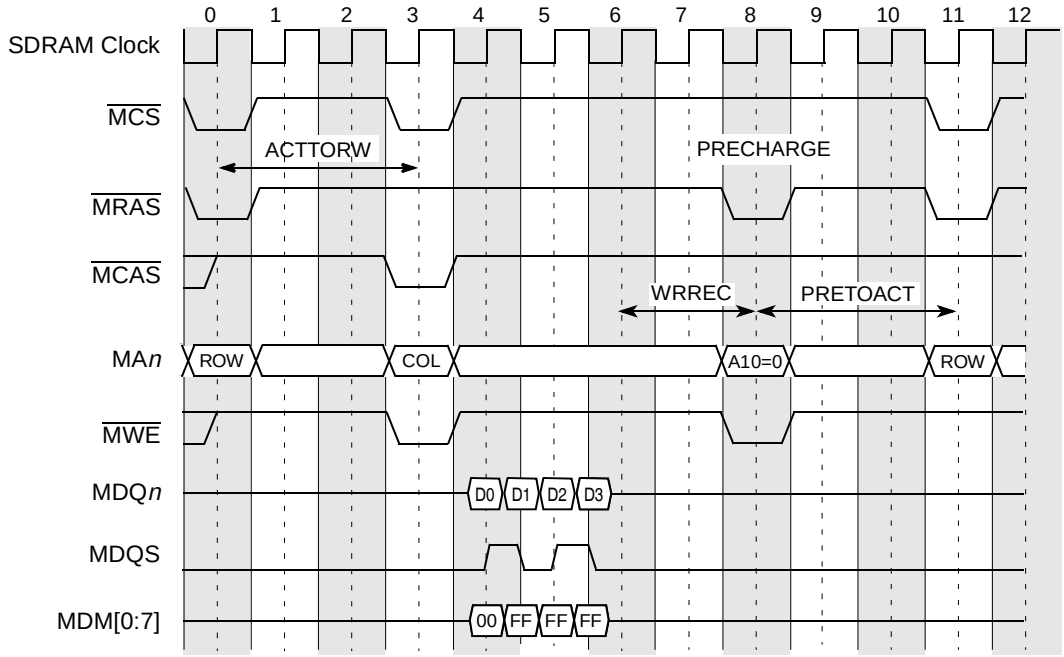


Figure 9-36. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

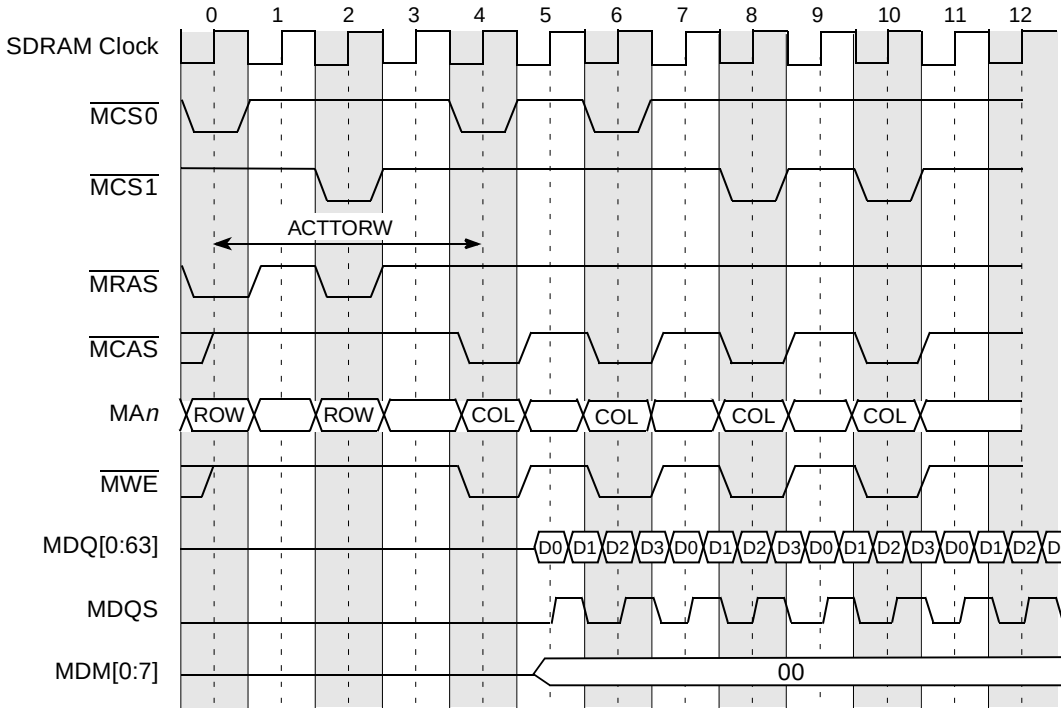


Figure 9-37. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

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.

- A 72 bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

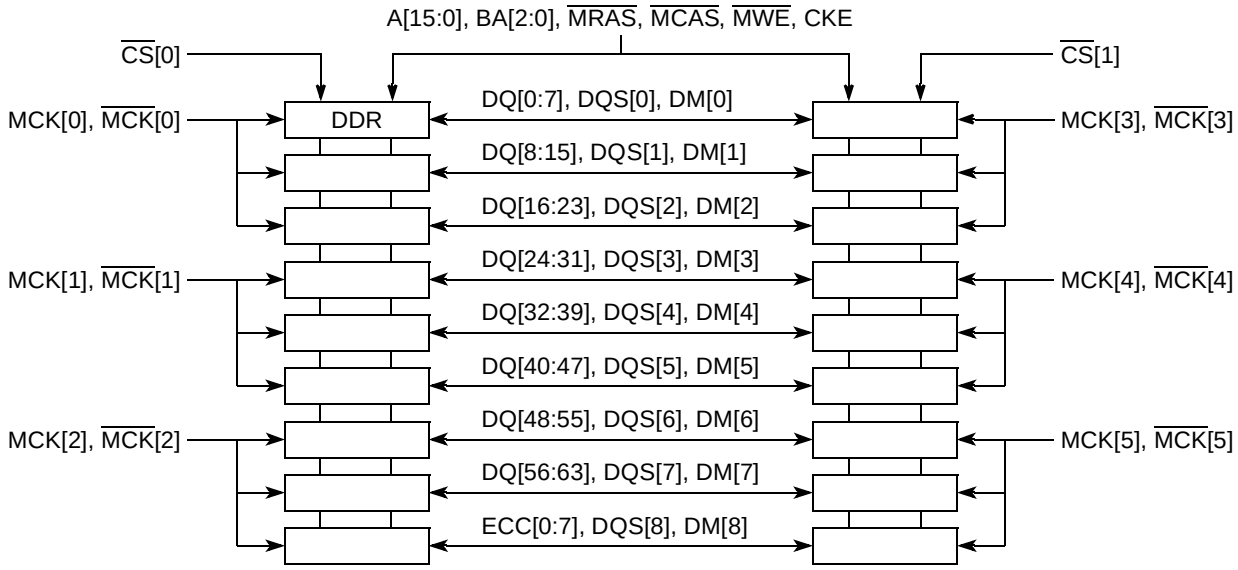


Figure 9-38. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

9.5.5 DDR SDRAM Mode-Set Command Timing

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

Figure 9-39 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

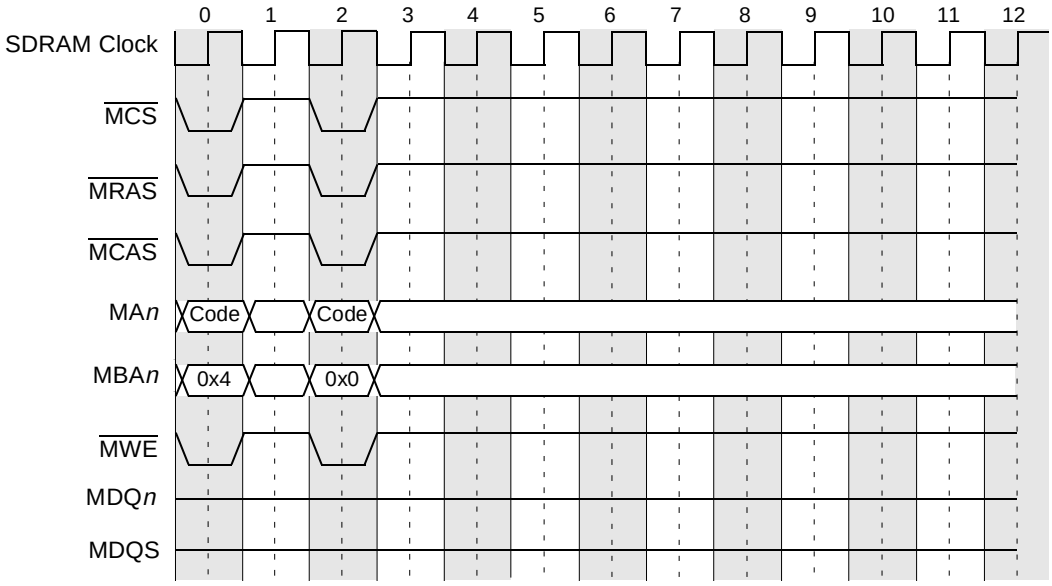


Figure 9-39. DDR SDRAM Mode-Set Command Timing

9.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting DDR_SDRAM_CFG[RD_EN] compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

Figure 9-40 shows the registered DDR SDRAM DIMM single-beat write timing.

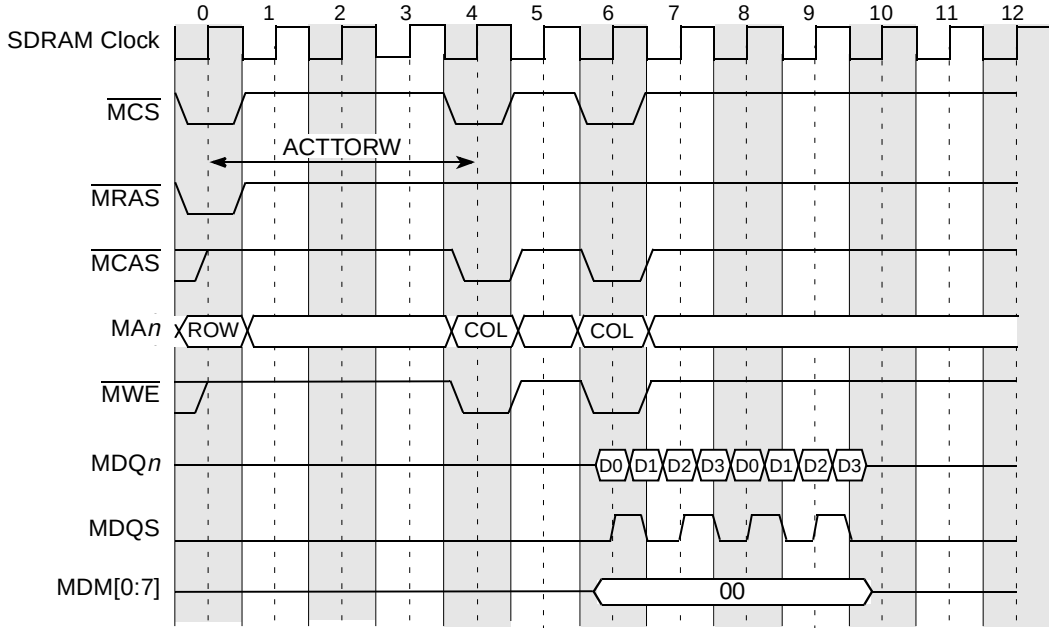


Figure 9-40. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMM. TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

Figure 9-41 shows the use of the WR_DATA_DELAY parameter.

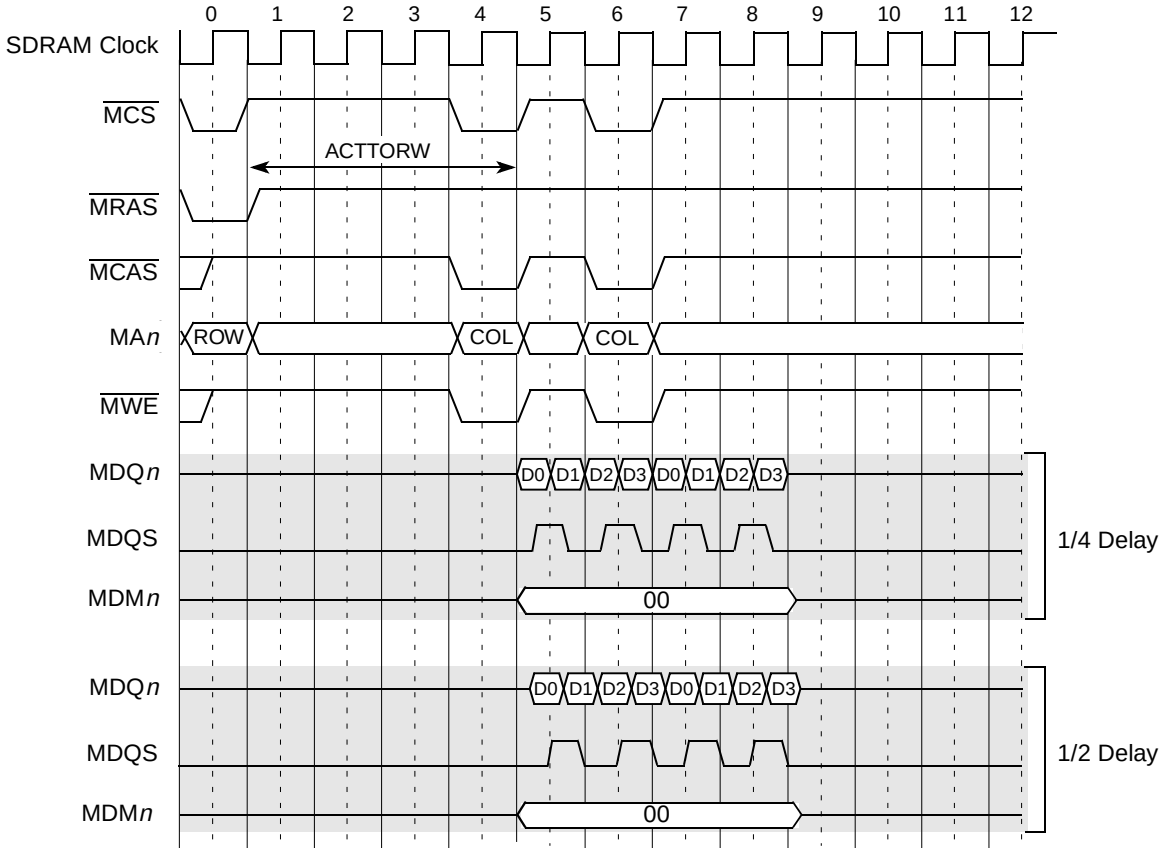


Figure 9-41. Write Timing Adjustments Example for Write Latency = 1

9.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle

waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

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

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

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

9.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in Figure 9-42 (TIMING_CFG_1 [REFREC] = 10 in this example).

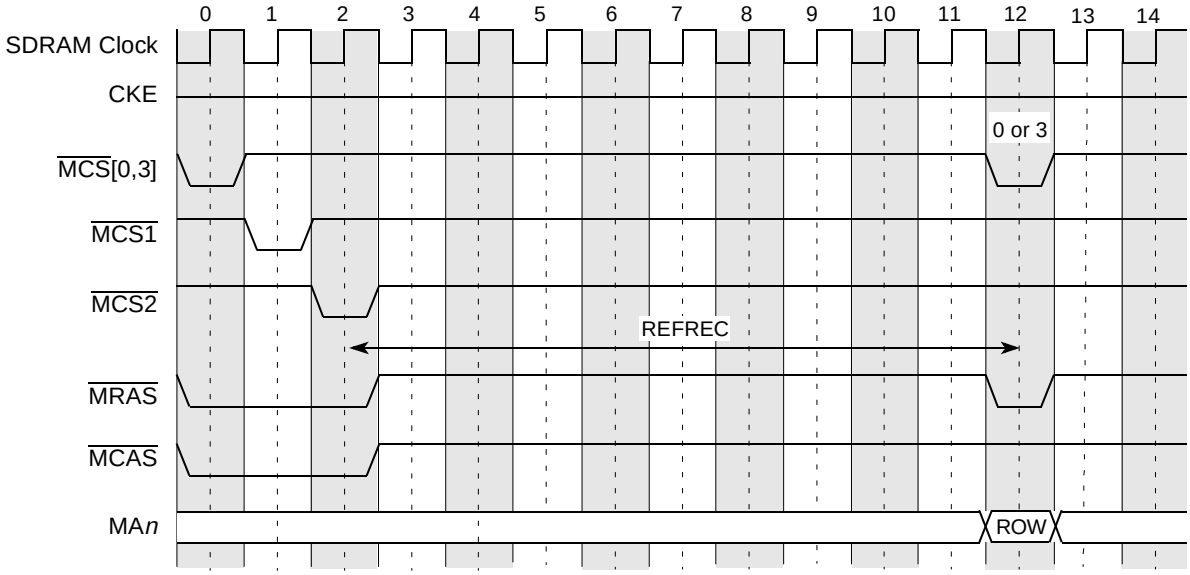


Figure 9-42. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

9.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 9-46 summarizes the refresh types available in each power-saving mode.

Table 9-46. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	—

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in Figure 9-43.

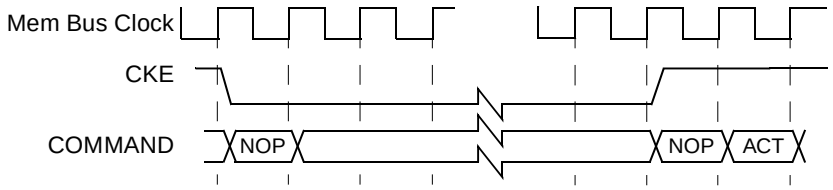


Figure 9-43. DDR SDRAM Power-Down Mode

9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-44 and Figure 9-45.

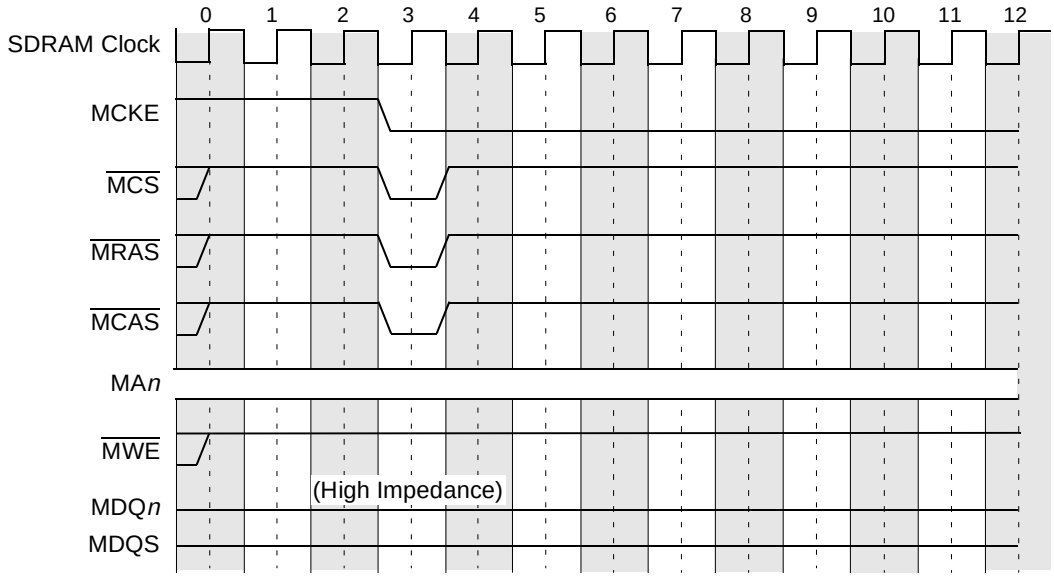


Figure 9-44. DDR SDRAM Self-Refresh Entry Timing

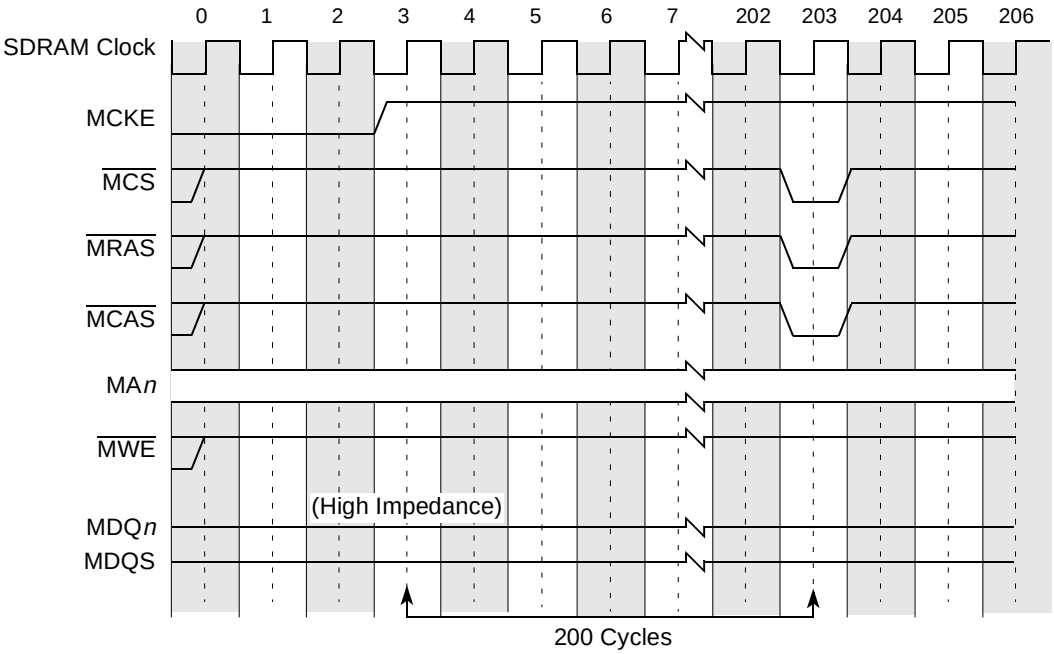


Figure 9-45. DDR SDRAM Self-Refresh Exit Timing

9.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not doubleword aligned or the size is not a multiple of a doubleword, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is doubleword aligned with a size that is a multiple of a doubleword, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full 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 9-47 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 9-47. Memory Controller–Data Beat Ordering

Transfer Size	Starting Double-Word Offset	Double-Word Sequence[1] to/from DRAM and Queues
1 double word	0 1 2 3	**0** - 1 - 2 - 3 **1** - 2 - 3 - 0 **2** - 3 - 0 - 1 **3** - 0 - 1 - 2
2 double words	0 1 2	**0 - 1** - 2 - 3 **1 - 2** - 3 - 0 **2 - 3** - 0 - 1
3 double words	0 1	**0 - 1 - 2** - 3 **1 - 2 - 3** - 0
4 double words	0 1 2 3	0 - 1 - 2 - 3 1 - 2 - 3 - 0 2 - 3 - 0 - 1 3 - 0 - 1 - 2

[1] All underlined double-word offsets are valid for the transaction.

9.5.10 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 by using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CSn_CONFIG[AP_n_EN].

9.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in Table 9-48 and Table 9-49.

Table 9-48. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
0	•	•						•	32			•	•				•
1	•		•					•	33			•		•			•
2	•			•				•	34	•		•		•			
3	•				•			•	35		•	•		•			
4	•	•				•			36			•	•		•		
5	•		•			•			37			•		•	•		
6	•			•		•			38	•		•		•	•		•
7	•				•	•			39		•	•		•	•		•
8	•	•					•		40			•	•			•	
9	•		•				•		41			•		•		•	
10	•			•			•		42	•		•		•		•	•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•				•	•	
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	
20		•	•			•			52		•				•		•
21		•		•		•			53			•			•		•
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•					•		•
24		•	•				•		56		•					•	•
25		•		•			•		57			•				•	•
26		•			•		•		58				•			•	•
27	•	•			•		•	•	59	•						•	•
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 9-49. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

9.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in Section 9.4.1.26, “Memory Error Interrupt Enable (ERR_INT_EN),” Section 9.4.1.25, “Memory Error Disable (ERR_DISABLE),” and Section 9.4.1.24, “Memory Error Detect (ERR_DETECT).”

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates a critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates the interrupt (if enabled, as described in Section 9.4.1.25, “Memory Error Disable (ERR_DISABLE)”). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a critical interrupt (if enabled, as described in Section 9.4.1.24, “Memory Error Detect (ERR_DETECT)”). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. Table 9-50 shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 9-50. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported via interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

9.6 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}n$ signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in Section 9.4.1.2, "Chip Select Configuration (CSn_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 9-51.

Table 9-51. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter		Section/page
CSn_BNDS	Chip select memory bounds	SAn EAn		9.4.1.1/9-11
CSn_CONFIG	Chip select configuration	CS_n_EN AP_n_EN ODT_RD_CFG ODT_WR_CFG	BA_BITS_CS_n ROW_BITS_CS_n COL_BITS_CS_n	9.4.1.2/9-12
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC		9.4.1.3/9-14
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	9.4.1.4/9-15
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	9.4.1.5/9-17
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE	WR_DATA_DELAY CKE_PLS FOUR_ACT	9.4.1.6/9-19

Table 9-51. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/page
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE NCAP 2T_EN BA_INTLV_CTL x32_EN HSE BI	9.4.1.7/9-21
DDR_SDRAM_CFG_2	Control configuration	SR_IE DLL_RST_DIS DQS_CFG ODT_CFG NUM_PR D_INIT	9.4.1.8/9-23
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE	9.4.1.9/9-25
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	9.4.1.10/9-26
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	9.4.1.12/9-29
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	9.4.1.13/9-30
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	9.4.1.14/9-30
DDR_INIT_ADDR	Initialization address	INIT_ADDR	9.4.1.15/9-31

9.6.1 Programming Differences Between Memory Types

Depending on the memory type used, certain fields must be programmed differently. Table 9-52 illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 9-52. Programming Differences Between Memory Types

Parameter	Description	Differences		Section/page
AP_*n*_EN	Chip Select *n* Auto Precharge Enable	DDR1	Can be used to place chip select *n* in auto precharge mode	9.4.1.2/9-12
		DDR2	Can be used to place chip select *n* in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR1	Should always be set to 000	9.4.1.2/9-12
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.	

Table 9-52. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR1	Should always be set to 000	9.4.1.2/9-12
		DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR1	Should be set to 0000	9.4.1.4/9-15
		DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD}.	
PRETOACT	Precharge to Activate Timing	DDR1	Should be set according to the specifications for the memory used (t_{RP})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RP})	
ACTTOPRE	Activate to Precharge Timing	DDR1	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	DDR1	Should be set according to the specifications for the memory used (t_{RCD})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RCD})	
CASLAT	CAS Latency	DDR1	Should be set, along with the Extended CAS Latency, to the desired CAS latency	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR1	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (t_{RFC})	9.4.1.5/9-17
		DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	
WRREC	Write Recovery	DDR1	Should be set according to the specifications for the memory used (t_{WR})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{WR})	
ACTTOACT	Activate *A* to Activate *B*	DDR1	Should be set according to the specifications for the memory used (t_{RRD})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{RRD})	

Table 9-52. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
WRTORD	Write to Read Timing	DDR1	Should be set according to the specifications for the memory used (t_{WRD})	9.4.1.5/9-17
		DDR2	Should be set according to the specifications for the memory used (t_{WRD})	
ADD_LAT	Additive Latency	DDR1	Should be set to 000	9.4.1.6/9-19
		DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR1	Should be set to 001	9.4.1.6/9-19
		DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then this field should be set to 100 (4 cycles).	
RD_TO_PRE	Read to Precharge Timing	DDR1	Should be set to 010 if burst length is 4 and 100 if burst length is 8	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR1	Can be set to 001	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	DDR1	Should be set to 0001	9.4.1.6/9-19
		DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.	
RD_EN	Registered DIMM Enable	DDR1	If registered DIMMs are used, then this field should be set to 1	9.4.1.7/9-21
		DDR2	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR1	If a 32-bit bus is used, and 8-beat bursts are desired, then this field should be set to 1	9.4.1.7/9-21
		DDR2	Should be set to 0	
2T_EN	2T Timing Enable	DDR1	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	9.4.1.7/9-21
		DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	

Table 9-52. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
DLL_RST_DIS	DLL Reset Disable	DDR1	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	9.4.1.8/9-23
		DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	
DQS_CFG	DQS Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Should be set to 00	
ODT_CFG	ODT Configuration	DDR1	Should be set to 00	9.4.1.8/9-23
		DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
BSTOPRE	Burst To Precharge Interval	DDR1	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	9.4.1.12/9-29
		DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

9.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller will conduct an automatic initialization sequence to the memory, which will follow the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

Chapter 10
Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers and contains a functional description of the general-purpose chip-select machine (GPCM), synchronous DRAM (SDRAM) machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes initialization and applications information sections with many specific examples of its use.

10.1 Introduction

Figure 10-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and an SDRAM controller.

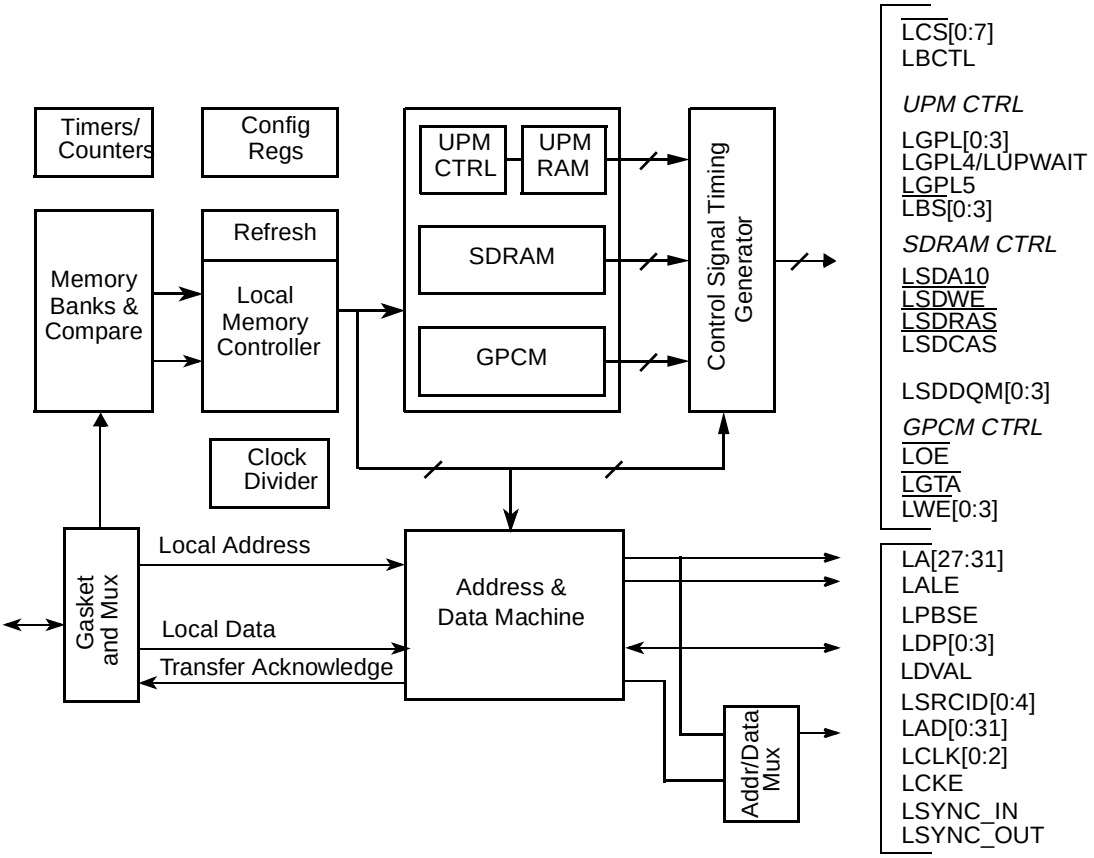


Figure 10-1. Local Bus Controller Block Diagram

The main component of the LBC is its memory controller, which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight

memory banks shared by a high performance SDRAM machine, a GPCM, and up to three UPMs. As such, it supports a minimal glue logic interface to SDRAM, SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LALE) allows multiplexing of addresses with data signals to reduce the device signal count.

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

10.1.1 Features

The LBC main features are as follows:

- Memory controller with eight memory banks
  - 32-bit address decoding with mask
  - Variable memory block sizes (32 Kbytes to 2 Gbytes)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - Automatic segmentation of large transactions
  - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
  - Write-protection capability
  - Atomic operation
  - Parity byte-select
- SDRAM machine
  - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
  - Supports up to 4 concurrent open pages per device
  - Supports SDRAM port size of 32-bit, 16-bit and 8-bit
  - Supports external address and/or command lines buffering
- General-purpose chip-select machine (GPCM)
  - Compatible with SRAM, EPROM, FEPROM, and peripherals
  - Global (boot) chip-select available at system reset
  - Boot chip-select support for 8-, 16-, or 32-bit devices
  - Minimum three-clock access to external devices
  - Four byte-write-enable signals ($\overline{\text{LWE}}$[0:3])
  - Output enable signal ($\overline{\text{LOE}}$)
  - External access termination signal ($\overline{\text{LGTA}}$)
- Three user-programmable machines (UPMs)
  - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period

  - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access.
  - UPM refresh timer runs a user-specified control signal pattern to support refresh
  - User-specified control-signal patterns can be initiated by software
  - Each UPM can be defined to support DRAM devices with depths of 64, 128, 256, and 512 Kbytes, and 1, 2, 4, 8, 16, 32, 64, 128, and 256 Mbytes
  - Support for 8-, 16-, 32-bit devices
  - Page mode support for successive transfers within a burst
  - Internal address multiplexing supporting 64-, 128-, 256-, and 512-Kbyte, and 1-, 2-, 4-, 8-, 16-, 32-, 64-, 128-, and 256-Mbyte page banks
- Optional monitoring of transfers between local bus internal masters and local bus slaves (local bus error reporting)
- Support for delay-locked loop (DLL) with software-configurable bypass for low frequency bus clocks

10.1.2 Modes of Operation

The LBC provides one GPCM, one SDRAM machine, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the LBC, the memory address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, SDRAM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus, with the LBC in GPCM, SDRAM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction. See Section 10.4, “Functional Description.”

10.1.2.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 2, 4, and 8 between the faster internal (system) clock and slower external bus clock (LCLK[0:2]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). This ratio has no functional effect on operation in SDRAM mode but it affects the resolution of signal timing shifts in GPCM mode and the interpretation of UPM array words in UPM mode. The bus clock is driven identically onto signals, LCLK[0:2] to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

10.1.2.2 Source ID Debug Mode

The LBC provides the ID of a transaction source on external device signals. When those signals are selected, the 5-bit internal ID of the current transaction source appears on LSRCID[0:4] whenever valid address or data is available on the LBC external signals. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID signals at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (LDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on LSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:31]. Note that in SDRAM mode the address vector contains the full address as {row, bank, column, lsb's} where row corresponds to the same row address for the given column address and lsb's are the unconnected lsb's of the address for a given port size.
- If a valid source ID is detected on LSRCID[0:4] and LDVAL is asserted, valid data may be latched from LAD[0:31].

The LSRCID[0:4] and LDVAL signals are multiplexed with other functions sharing the same external signals. Refer to chapter 3, external signals description and to chapter 5, system configuration to learn how to enable the LSRCID/LDVAL signals.

10.2 External Signal Descriptions

Table 10-1 contains a list of external signals related to the LBC and summarizes their function. The table also shows the reset state of all external signals during assertion of $\overline{\text{HRESET}}$. For more information on the use of some of these signals as reset configuration signals on the MPC8360E, see Section 4.2.2, "Power-On Reset Flow." Note that during assertion of $\overline{\text{HRESET}}$, the DLL is initially unlocked, so the LCLK and LSYNC_OUT values are likely to be unstable/jittery for several microseconds; after the DLL locks, stable clock signals are driven on these signals.

Table 10-1. Signal Properties—Summary

Name	Alternate Functions	Mode	Descriptions	No. of Signals	I/O	Reset State (Outputs)
LALE	—	—	External address latch enable	1	O	Reset_cfg
$\overline{\text{LCS0}}$	—	—	Chip select 0	1	O	Reset_cfg
LCS[1:7]	—	—	Chip selects [1–7]	7	O	All high
$\overline{\text{LWE}}$[0:3]/ LSDDQM/ LBS[0:3]	$\overline{\text{LWE}}$[0:3]	GPCM	Write enable	4	O	Reset_cfg
	SDDQM	SDRAM	Byte lane data mask			
	$\overline{\text{LBS}}$[0:3]	UPM	Byte (lane) select			
LGPL0 LSDA10	LGPL0	UPM	General purpose line 0	1	O	Reset_cfg
	LSDA10	SDRAM	Row address bit/command bit			
LGPL1 $\overline{\text{LSDWE}}$	LGPL1	UPM	General purpose line 1	1	O	Reset_cfg
	LSDWE	SDRAM	Write enable			
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	LOE	GPCM	Output enable	1	O	
	LSDRAS	SDRAM	Row address strobe			
	LGPL2	UPM	General purpose line 2			
LGPL3 $\overline{\text{LSDCAS}}$	LGPL3	UPM	General purpose line 3	1	O	Reset_cfg
	LSDCAS	SDRAM	Column address strobe			

Table 10-1. Signal Properties—Summary (continued)

Name	Alternate Functions	Mode	Descriptions	No. of Signals	I/O	Reset State (Outputs)
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT/ LPBSE	LGTA	GPCM	Transaction termination	1	I	High-Z
	LGPL4	UPM	General purpose line 4		O	
	LUPWAIT	UPM	External device wait		I	
	LPBSE	—	Local bus parity byte select		O	
LGPL5	—	UPM	General purpose line 5	1	O	Reset_cfg
LBCTL	—	—	Data buffer control	1	O	
LA[27:31]	—	—	Local bus non-multiplexed address lsb's	5	O	
LAD[0:31]	—	—	Multiplexed address/data bus	32	I/O	
LDP	—	—	Local bus data parity	4	I/O	High-Z
LCKE	—	—	Local bus clock enable	1	O	High
LCLK[0:2]	—	—	Local bus clocks	3	O	Driven
LSYNC_IN	—	—	DLL synchronize input	1	I	—
LSYNC_OUT	—	—	DLL synchronize output	1	O	Driven
LDVAL	—	LBC debug	Local bus data valid	1	O	Not connected to external signals
LSRCID[0:4]	—	LBC debug	Local bus source ID	5	O	Not connected to external signals

Table 10-2 shows the detailed external signal descriptions for the LBC.

Table 10-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description	
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device signals. See Section 10.4.1.2, "External Address Latch Enable Signal (LALE)."	
		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.
$\overline{\text{LCS}}$[0:7]	O	Chip selects. Eight chip selects are provided which are mutually exclusive.	
		State Meaning	Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. $\overline{\text{LCS}}$[0:7] are provided on a per-bank basis with $\overline{\text{LCS0}}$ corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LWE}}$[0:3]/ LSDDQM[0:3]/ $\overline{\text{LBS}}$[0:3]	O	GPCM write enable/SDRAM data mask/UPM byte select. These signals select or validate each byte lane of the data bus. For banks with port sizes of 32 bits (as set by BR*n*[PS]), all four signals are defined. For a 16-bit port size, only bits 0–1 are defined; and for an 8-bit port size, bit 0 is the only defined signal. The least-significant address bits of each access also determine which byte lanes are considered valid for a given data transfer.	
		State Meaning	Asserted/Negated—For GPCM operation, $\overline{\text{LWE}}$[0:3] assert for each byte lane enabled for writing. For SDRAM operation, LSDDQM[0:3] function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. LSDDQM[0:3] are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. $\overline{\text{LBS}}$[0:3] are programmable byte-select signals in UPM mode. See Section 10.4.4.4, "UPM RAM Array," for programming details about $\overline{\text{LBS}}$[0:3].
		Timing	Assertion/Negation—See Section 10.4.2, "General-Purpose Chip-Select Machine (GPCM)," for details regarding the timing of $\overline{\text{LWE}}$[0:3].
LSDA10/ LGPL0	O	SDRAM A10/General purpose line 0	
		State Meaning	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general purpose signals when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDWE}}$/ LGPL1	O	SDRAM write enable/General-purpose line 1	
		State Meaning	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	O	GPCM output enable/SDRAM RAS/General-purpose line 2	
		State Meaning	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe (RAS). One of six general purpose lines when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDCAS}}$/ LGPL3	O	SDRAM CAS/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe (CAS). This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT/ LPBSE	I/O	GPCM transfer acknowledge/General-purpose line 4/UPM wait/parity byte select	
		State Meaning	Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing $\overline{\text{LBS}}$[0:3] through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC provides this optional byte-select signal that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to n -parity devices. **Note:** The $\overline{\text{LGTA}}$/LUPWAIT signal can be sampled as asserted (active-low) during LGPL4/LPBSE is brought low by the UPM. For a subsequent GPCM transaction, it functions as $\overline{\text{LGTA}}$/LUPWAIT. As a result, GPCM transactions may be terminated prematurely before $\overline{\text{LGTA}}$/LUPWAIT has drifted to a logical one. **Work Around:** One way to resolve this issue is to ensure that $\overline{\text{LGTA}}$/LGPL4 is pulled-up to 3.3 V by an external 1-KΩ register. This will ensure that $\overline{\text{LGTA}}$ is sampled high (not asserted) by the time any subsequent GPCM transactions are initiated by the local bus memory controller. If this signal is used purely as an input ($\overline{\text{LGTA}}$/LUPWAIT), a weaker (10-KΩ) pull-up register may be substituted; alternatively, if this signal is used as LPBSE, no pull-up is required, as $\overline{\text{LGTA}}$/LUPWAIT are disabled. 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. Access to an SDRAM machine-controlled bank does not activate the buffer control. Buffer control is disabled by setting OR*n*[BCTLD].	
		State Meaning	Asserted/Negated—Normally functions as a write/$\overline{\text{read}}$ control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
LA[27:31]	O	Local bus nonmultiplexed address lsb's. All bits driven on LA[27:31] are defined for 8-bit port sizes. For 32-bit port sizes, LA[30:31] are don't cares; for 16-bit port sizes LA31 is a don't care.	
		State Meaning	Asserted/Negated—Although the LBC shares an address and data bus, up to five lsbs of the RAM address always appear on the dedicated address signals, LA[27:31]. These may be used, unlatched, in place of LAD[27:31] to connect the five lsbs of the address for address phases. For some RAM devices, such as fast-page DRAM, LA[27:31] serve as the column address offset during a burst access.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in BR*n*[PS] as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). For a port size of 16 bits, LAD[0:7] connect to the most significant byte lane (at address offset 0), while LAD[8:15] connect to the least significant byte lane (at address offset 1); LAD[16:31] are unused for 16-bit port sizes. For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.	
		State Meaning	Asserted/Negated—LAD[0:31] is the shared 32-bit address/data bus through which external RAM devices transfer data and receive addresses
		Timing	Assertion/Negation—During assertion of LALE, LAD[0:31] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:31] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:31] are either driven by write data or are made high-impedance by the LBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD[0:31] are again taken into a high-impedance state.
LDP[0:3]	I/O	Local bus data parity. Drives and receives the data parity corresponding with the data phases on LAD[0:31].	
		State Meaning	Asserted/Negated—During write accesses, a parity bit is generated for each 8 bits of LAD[0:31], such that LDP0 is even/odd parity for LAD[0:7], while LDP[3] is even/odd parity for LAD[24:31]. Unused byte lanes for port sizes less than 32 bits have undefined parity.
		Timing	Assertion/Negation—Drive and receive the data parity corresponding with the data phases on LAD[0:31]. For read accesses, the parity bits for each byte lane are sampled on LDP[0:3] with the same timing that read data is sampled on LAD[0:31]. LDP[0:3] change impedance in concert with LAD[0:31].
LCKE	O	Local bus clock enable	
		State Meaning	Asserted/Negated—Bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. Asserted during normal SDRAM operation.
LCLK[0:2]	O	Local bus clocks. Enabled by the DLLCK register. See Section 18.4.3, “DLL Clock Register (DLLCK),” for details.	
		State Meaning	Asserted/Negated—LCLK[0:2] drive an identical bus clock signal for distributed loads. If the LBC DLL is enabled (see LCRR[DBYP], Figure 10-19 on page 10-31), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:310:15] and $\overline{\text{LCS}n}$) by a time delay matching the delay of the DLL timing loop set up between LSYNC_OUT and LSYNC_IN. DLLLCK
LSYNC_OUT	O	DLL synchronization out.	
		State Meaning	Asserted/Negated—A replica of the bus clock, appearing on LSYNC_OUT, should be propagated through a passive timing loop and returned to LSYNC_IN for achieving correct DLL lock.
		Timing	Assertion/Negation—The time delay of the timing loop should be such that it compensates for the round-trip flight time of LCLK[0:2] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	DLL synchronization in.	
		State Meaning	Asserted/Negated—See description of LSYNC_OUT.

Table 10-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LDVAL	O	Local bus data valid (LBC debug mode only)	
		State Meaning	Asserted/Negated—For a read, LDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:31]. For a write, LDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, LDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, LDVAL asserts when the LBC generates a data transfer acknowledge.
LSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all LSRCID[0:4] signals are driven high unless LSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until LDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, LSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC. For example, LSRCID[0:4] is valid only during CAS phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a CAS cycle.

10.3 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]	10.3.1.1/10-11
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			

Table 10-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5004	OR0—Option register 0	R/W	0x0000_0FF7	10.3.1.2/10-12
0x0_500C	OR1—Option register 1		0x0000_0000	
0x0_5014	OR2—Option register 2			
0x0_501C	OR3—Option register 3			
0x0_5024	OR4—Option register 4			
0x0_502C	OR5—Option register 5			
0x0_5034	OR6—Option register 6			
0x0_503C	OR7—Option register 7			
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	10.3.1.3/10-18
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	10.3.1.4/10-19
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	10.3.1.5/10-21
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	10.3.1.6/10-22
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	10.3.1.7/10-22
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	10.3.1.8/10-24
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	10.3.1.9/10-25
0x0_50B0	LTESR—Transfer error status register	Read/bit-reset	0x0000_0000	10.3.1.10/10-25
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	10.3.1.11/10-27
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	10.3.1.12/10-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	10.3.1.13/10-29
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	10.3.1.14/10-29
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	10.3.1.15/10-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	10.3.1.16/10-31

[1] Port size for BR0 is configured from the value of RCWH[ROMLOC], which is loaded during reset, hence ‘RR’ is either 0x08, 0x10, or 0x18.

10.3.1 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–BR7)

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]–BR7[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) 0x0_5020 (BR4) Access: read/write
0x0_5008 (BR1) 0x0_5028 (BR5)
0x0_5010 (BR2) 0x0_5030 (BR6)
0x0_5018 (BR3) 0x0_5038 (BR7)

	0														16	17	18	19	20	21	22	23	24		26	27	28		30	31
R W	BA														—			PS	DECC			WP	MSEL			—	ATOM		—	V
Reset[1]	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 32-bit
21–22	DECC	Specifies the method for data error checking. 00 Data error checking disabled, but normal parity generation 01 Normal parity generation and checking 10 Read-modify-write parity generation and normal parity checking (32-bit port size only) 11 Reserved

Table 10-4. BR*n* Field Descriptions (continued)

Bits	Name	Description
23	WP	Write protect. 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert $\overline{\text{LCS}n}$ on write cycles to this memory bank. LTESR[WP] is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26	MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (reset value) 001 Reserved 010 Reserved 011 SDRAM 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	—	Reserved
28–29	ATOM	Atomic operation. Writes (reads) to the address space handled by the memory controller bank reserve the selected memory bank for the exclusive use of the accessing device. The reservation is released when the device performs a read (write) operation to this memory controller bank. If a subsequent read (write) request to this memory controller bank is not detected within 256 bus clock cycles of the last write (read), the reservation is released and an atomic error is reported (if enabled). 00 The address space controlled by this bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA). 10 Write-after-read-atomic (WARA). 11 Reserved
30	—	Reserved
31	V	Valid bit. Indicates that the contents of the BR*n* and OR*n* pair are valid. $\overline{\text{LCS}n}$ does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

10.3.1.2 Option Registers (OR0–OR7)

The OR*n* registers define the sizes of memory banks and access attributes. The OR*n* attribute bits support the following three modes of operation as defined by BR*n*[MSEL].

- GPCM mode (see Section 10.3.1.2.2, “Option Registers (ORn)—GPCM Mode.”)
- UPM mode (see Section 10.3.1.2.3, “Option Registers (ORn)—UPM Mode.”)
- SDRAM mode (see Section 10.3.1.2.4, “Option Registers (ORn)—SDRAM 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 4 Gbytes.

Table 10-5. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

10.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 10-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

Offset 0x0_5004 (OR0) 0x0_5024 (OR4) Access: read/write
0x0_500C (OR1) 0x0_502C (OR5)
0x0_5014 (OR2) 0x0_5034 (OR6)
0x0_501C (OR3) 0x0_503C (OR7)

	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	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

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–16</td><td>AM</td><td>GPCM address mask. Masks corresponding BRn 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.”</td></tr>
<tr><td>17–18</td><td>—</td><td>Reserved</td></tr>
<tr><td>19</td><td>BCTLD</td><td>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.</td></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 rowspan="2">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>01</td><td>Reserved.</td></tr>
<tr><td rowspan="2">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>11</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
<tr><td rowspan="2">4 or 8</td><td>10</td><td>$\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines.</td></tr>
<tr><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 10-23 and Table 10-24 for LCRR[CLKDIV] = 4 or 8, Table 10-25 and Table 10-26 for LCRR[CLKDIV] = 2).</td></tr>
</table>

Table 10-6. OR*n*—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></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>
<tr><td>29</td><td>TRLX</td><td>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.</td></tr>
<tr><td>30</td><td>EHTR</td><td>Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
<table>
<tr><th>TRLX</th><th>EHTR</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>The memory controller generates normal timing. No additional cycles are inserted.</td></tr>
<tr><td>0</td><td>1</td><td>1 idle clock cycle is inserted.</td></tr>
<tr><td>1</td><td>0</td><td>4 idle clock cycles are inserted.</td></tr>
<tr><td>1</td><td>1</td><td>8 idle clock cycles are inserted.</td></tr>
</table></td></tr>
<tr><td>31</td><td>EAD</td><td>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]).</td></tr>
</table>

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), 0x0_500C (OR1), 0x0_5014 (OR2), 0x0_501C (OR3), 0x0_5024 (OR4), 0x0_502C (OR5), 0x0_5034 (OR6), 0x0_503C (OR7) Access: read/write

	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.

Table 10-7. OR*n*—UPM Field Descriptions (continued)

Bits	Name	Description
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access. TRLX / EHTR / Meaning 0 / 0 / The memory controller generates normal timing. No additional cycles are inserted. 0 / 1 / 1 idle clock cycle is inserted. 1 / 0 / 4 idle clock cycles are inserted. 1 / 1 / 8 idle clock cycles are inserted.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

10.3.1.2.4 Option Registers (OR*n*)—SDRAM Mode

Figure 10-5 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the SDRAM machine.

Offset 0x0_5004 (OR0) 0x0_5024 (OR4) Access: Read/write
0x0_500C (OR1) 0x0_502C (OR5)
0x0_5014 (OR2) 0x0_5034 (OR6)
0x0_501C (OR3) 0x0_503C (OR7)

	0–16	17–18	19–21	22	23–25	26–27	30	31
R W	AM	—	COLS	—	ROWS	PMSEL	—	EAD
Reset	All zeros							

Figure 10-5. Option Registers (OR*n*) in SDRAM Mode

Table 10-8 describes OR*n* fields for SDRAM mode.

Table 10-8. OR*n*—SDRAM Field Descriptions

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. AM can be read or written at any time. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address signals.
17–18	—	Reserved

Table 10-8. OR*n*—SDRAM Field Descriptions (continued)

Bits	Name	Description
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7 100 11 001 8 101 12 010 9 110 13 011 10 111 14
22	—	Reserved
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 100 13 001 10 101 14 010 11 110 15 011 12 111 Reserved
26	PMSEL	Page mode select. Selects page mode for the SDRAM connected to the memory controller bank. 0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.
27–30	—	Reserved
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

10.3.1.3 UPM Memory Address Register (MAR)

Figure 10-6 shows the fields of the UPM memory address register (MAR).

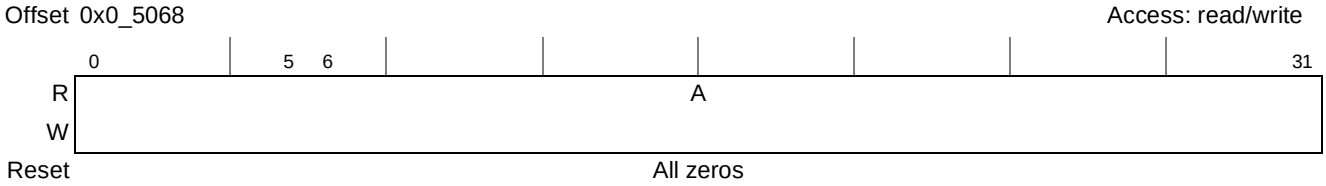


Figure 10-6. UPM Memory Address Register (MAR)

Table 10-9 describes the MAR fields.

Table 10-9. MAR Field Descriptions

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

10.3.1.4 UPM Mode Registers (M*n*MR)

The UPM machine mode registers (MAMR, MBMR, and MCMR), shown in Figure 10-7, contain the configuration for the three UPMs.

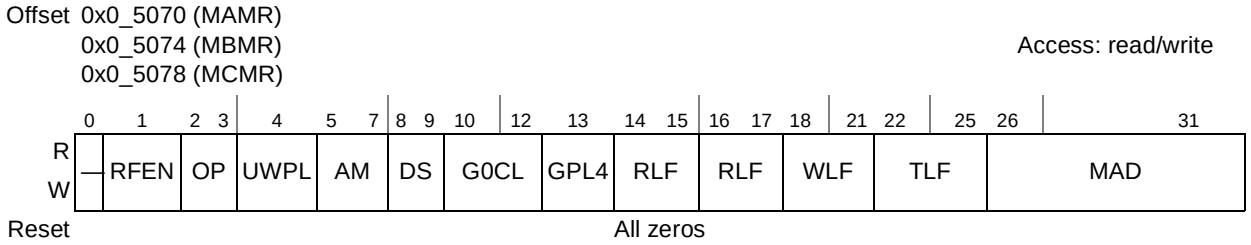


Figure 10-7. UPM Mode Registers (M*n*MR)

Table 10-10 describes UPM mode fields.

Table 10-10. 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 10-10. 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–LA15</th><th>LA16</th><th>LA17</th><th>LA18</th><th>LA19–LA28</th><th>LA29</th><th>LA30</th><th>LA31</th></tr>
<tr><td>000</td><td>0</td><td>A8</td><td>A9</td><td>A10</td><td>A11–A20</td><td>A21</td><td>A22</td><td>A23</td></tr>
<tr><td>001</td><td>0</td><td>A7</td><td>A8</td><td>A9</td><td>A10–A19</td><td>A20</td><td>A21</td><td>A22</td></tr>
<tr><td>010</td><td>0</td><td>A6</td><td>A7</td><td>A8</td><td>A9–A18</td><td>A19</td><td>A20</td><td>A21</td></tr>
<tr><td>011</td><td>0</td><td>A5</td><td>A6</td><td>A7</td><td>A8–A17</td><td>A18</td><td>A19</td><td>A20</td></tr>
<tr><td>100</td><td>0</td><td>A4</td><td>A5</td><td>A6</td><td>A7–A16</td><td>A17</td><td>A18</td><td>A19</td></tr>
<tr><td>101</td><td>0</td><td>A3</td><td>A4</td><td>A5</td><td>A6–A15</td><td>A16</td><td>A17</td><td>A18</td></tr>
<tr><td>110–111</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 10-30 on page 10-68.
<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 10-10. 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 10-8, is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

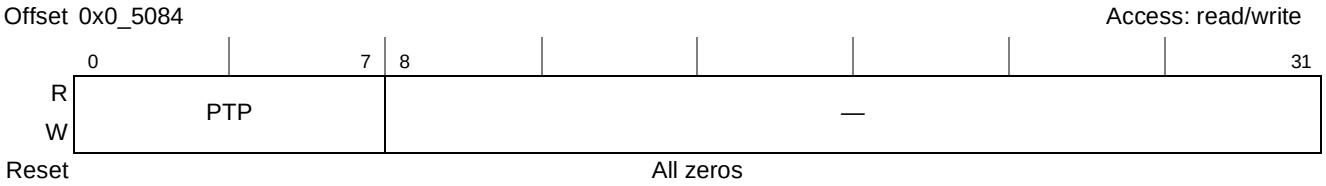


Figure 10-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 10-11 describes MRTPR fields.

Table 10-11. MRTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 0x0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

10.3.1.6 UPM Data Register (MDR)

MDR shown in Figure 10-9, contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.

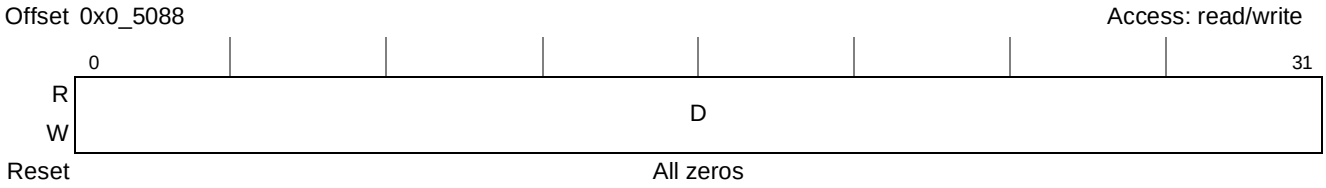


Figure 10-9. UPM Data Register (MDR)

Table 10-12 describes MDR[D].

Table 10-12. MDR Field Description

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM (M*n*MR[OP] = 01 or M*n*MR[OP] = 10).

10.3.1.7 Local Bus SDRAM Machine Mode Register (LSDMR)

LSDMR shown in Figure 10-10, is used to configure operations pertaining to SDRAM.

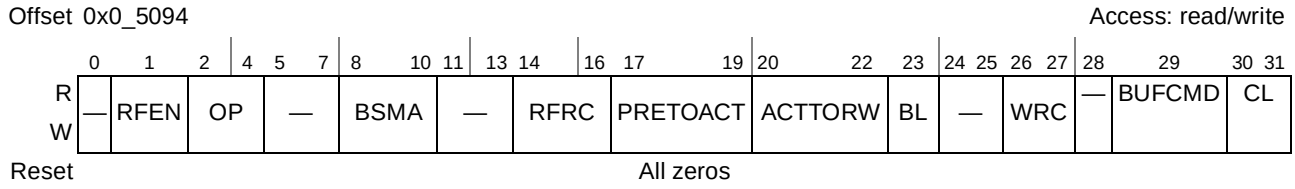


Figure 10-10. Local Bus SDRAM Machine Mode Register (LSDMR)

Table 10-12 describes LSDMR fields.

Table 10-13. LSDMR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the SDRAM requires refresh services. 0 Refresh services are not required. 1 Refresh services are required.

Table 10-13. LSDMR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–4</td><td>OP</td><td>SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed. See Section 10.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands.”
<table>
<tr><th>Value</th><th>Meaning</th><th>Use</th></tr>
<tr><td>000</td><td>Normal operation</td><td>Normal operation</td></tr>
<tr><td>001</td><td>Auto refresh</td><td>Initialization</td></tr>
<tr><td>010</td><td>Self refresh</td><td>Debug</td></tr>
<tr><td>011</td><td>Mode register write</td><td>Initialization</td></tr>
<tr><td>100</td><td>Precharge bank</td><td>Debug</td></tr>
<tr><td>101</td><td>Precharge all banks</td><td>Initialization</td></tr>
<tr><td>110</td><td>Activate bank</td><td>Debug</td></tr>
<tr><td>111</td><td>Read/write without valid data transfer</td><td>Debug</td></tr>
</table></td></tr>
<tr><td>5–7</td><td>—</td><td>Reserved</td></tr>
<tr><td>8–10</td><td>BSMA</td><td>Bank select multiplexed address line. Selects which address signals serve as the 2-bit bank-select address for SDRAM. Note that only 4-bank SDRAMs are supported.

000 LA[12:13] 100 LA[16:17]
001 LA[13:14] 101 LA[17:18]
010 LA[14:15] 110 LA[18:19]
011 LA[15:16] 111 LA[19:20]</td></tr>
<tr><td>11–13</td><td>—</td><td>Reserved</td></tr>
<tr><td>14–16</td><td>RFRC</td><td>Refresh recovery. Sets the refresh recovery interval in bus clock cycles. Defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command. See Section 10.4.3.7.5, “Refresh Recovery Interval (RFRC).”

000 Reserved 100 6 clocks
001 3 clocks 101 7 clocks
010 4 clocks 110 8 clocks
011 5 clocks 111 16 clocks</td></tr>
<tr><td>17–19</td><td>PRETOACT</td><td>Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states). See Section 10.4.3.7.1, “Precharge-to-Activate Interval.”

000 8 100 4
001 1 101 5
010 2 110 6
011 3 111 7</td></tr>
<tr><td>20–22</td><td>ACTTORW</td><td>Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). See Section 10.4.3.7.2, “Activate-to-Read/Write Interval.”

000 8 100 4
001 Reserved 101 5
010 2 110 6
011 3 111 7</td></tr>
</table>

Table 10-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
23	BL	Sets the burst length for SDRAM accesses. 0 SDRAM burst length is 4. Use this value if the device port size is 16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
24–25	—	Reserved
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data is written to the SDRAM. See Section 10.4.3.7.3, "Column Address to First Data Out—CAS Latency." 00 4 01 Reserved 10 2 11 3
28	—	Reserved
29	BUFCMD	Control line assertion timing. If external buffers are placed on the control lines going to both the SDRAM and address lines, setting BUFCMD causes all SDRAM control lines except $\overline{\text{LCS}n}$, LCKE, LALE and $\overline{\text{LSDDQM}[0:3]}$ to be asserted for LCRR[BUFCMDC] cycles, instead of one. See Section 10.4.3.7.6, "External Address and Command Buffers (BUFCMD)." 0 Normal timing for the control lines 1 All control lines except $\overline{\text{LCS}n}$ are asserted for the number of cycles specified by LCRR[BUFCMDC].
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. 00 Extended CAS latency. According to LCRR[ECL]. See Table 10-22. 01 1 10 2 11 3

10.3.1.8 UPM Refresh Timer (LURT)

LURT, shown in Figure 10-11, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled (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.

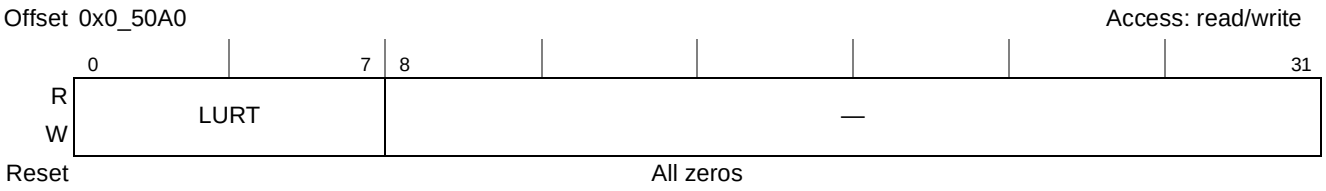


Figure 10-11. UPM Refresh Timer (LURT)

Table 10-14 describes LURT fields.

Table 10-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LURT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ Example: For a 266-MHz system clock and a required service rate of 15.6 µs, given MRTPR[PTP] = 32, the LURT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value (0x00) sets the maximum period to 256 x MRTPR[PTP] system clock cycles.
8–31	—	Reserved

10.3.1.9 SDRAM Refresh Timer (LSRT)

LSRT shown in Figure 10-12, generates a refresh request for all valid banks that selected a SDRAM machine and are refresh-enabled (LSDMR[RFEN] = 1). Each time the timer expires, all qualifying banks generate a bank staggering auto-refresh request using the SDRAM machine.

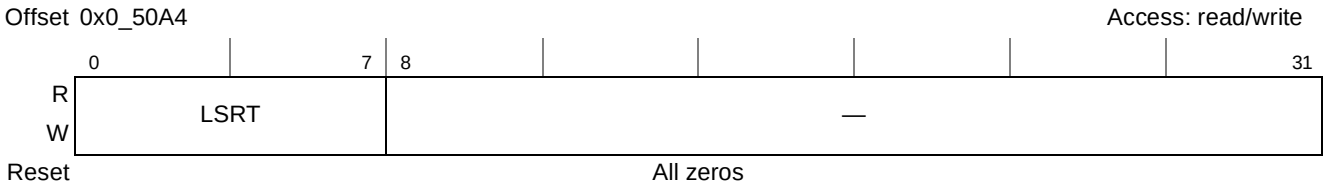


Figure 10-12. LSRT SDRAM Refresh Timer (LSRT)

Table 10-15 describes LSRT fields.

Table 10-15. LSRT Field Descriptions

Bits	Name	Description
0–7	LSRT	SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ **Example:** For a 266-MHz system clock and a required service rate of 15.6 µs, given PTP = 32, the LSRT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value, 0x00, sets the maximum period to 256 x MRTPR[PTP] system clock cycles.
8–31	—	Reserved

10.3.1.10 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 10-13, 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.

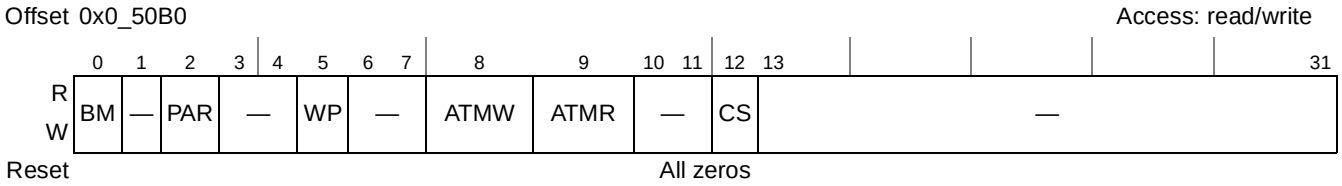


Figure 10-13. Transfer Error Status Register (LTESR)

Table 10-16 describes LTESR fields.

Table 10-16. LTESR Field Descriptions

Bits	Name	Description
0	BM	Bus monitor time-out. 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[BMT] x 8 bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	Parity 0 No local bus parity error. 1 Local bus parity error. LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BNK] indicates which memory controller bank was accessed.
3–4	—	Reserved
5	WP	Write protect error 0 No write protect error occurred. 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur (as the cycle is not automatically terminated).
6–7	—	Reserved
8	ATMW	Atomic error 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

Table 10-16. LTESR Field Descriptions (continued)

Bits	Name	Description
12	CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the LBC that did not hit any memory bank.
13–31	—	Reserved

10.3.1.11 Transfer Error Check Disable Register (LTEDR)

LTEDR shown in Figure 10-14, is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) through the interrupt mechanism.

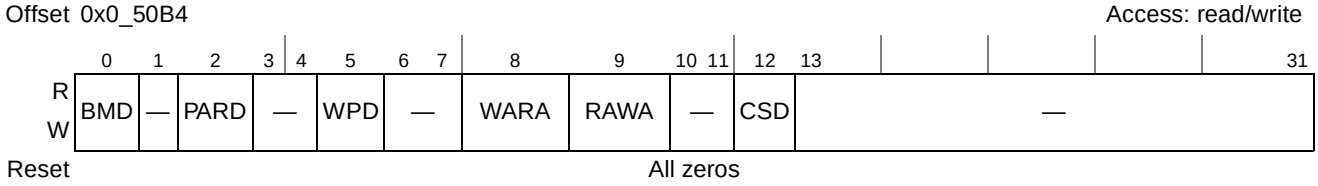


Figure 10-14. Transfer Error Check Disable Register (LTEDR)

Table 10-17 describes LTEDR fields.

Table 10-17. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable. 0 Bus monitor is enabled. 1 Bus monitor is disabled.
1	—	Reserved
2	PARD	Parity error checking disabled. 0 Parity error checking is enabled. 1 Parity error checking is disabled.
3–4	—	Reserved
5	WPD	Write protect error checking disable. 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write after read atomic (WARA) error checking disable. 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read after write atomic (RAWA) error checking disable. 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved

Table 10-17. LTEDR Field Descriptions (continued)

Bits	Name	Description
12	CSD	Chip select error checking disable. 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

10.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

LTEIR shown in Figure 10-15, is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.

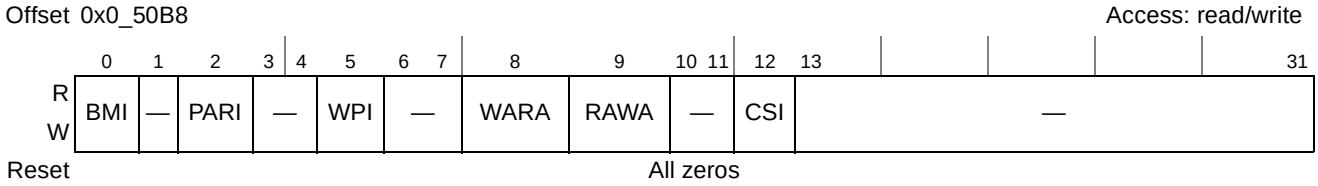


Figure 10-15. Transfer Error Interrupt Enable Register (LTEIR)

Table 10-18 describes LTEIR fields.

Table 10-18. LTEIR Field Descriptions

Bits	Name	Description
0	BMI	Bus monitor error interrupt enable. 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1	—	Reserved
2	PARI	Parity error interrupt enable. 0 Parity error reporting is disabled. 1 Parity error reporting is enabled.
3–4	—	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
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

Table 10-18. LTEIR Field Descriptions (continued)

Bits	Name	Description
12	CSI	Chip select error interrupt enable. 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

10.3.1.13 Transfer Error Attributes Register (LTEATR)

LTEATR is shown in Figure 10-16. After LTEATR[V] has been set, software must clear this bit to allow LTEATR and LTEAR to update following any subsequent errors.

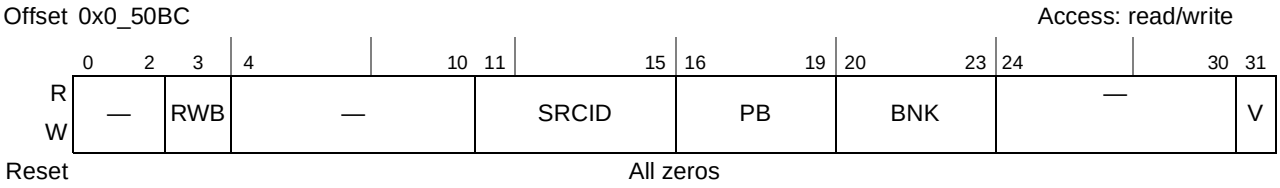


Figure 10-16. Transfer Error Attributes Register (LTEATR)

Table 10-19 describes LTEATR fields.

Table 10-19. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	RWB	Transaction type for the error: 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–10	—	Reserved
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the LBC. The coding of the source ID debug information is the same as the coding of AEATR[MSTR_ID] (see Section 6.2.6, “Arbiter Event Attributes Register (AEATR).”)
16–19	PB	Parity error on byte. There are four parity error status bits, one per byte lane. A bit is set for the byte that had a parity error (bit 16 represents byte 0, the most significant byte lane).
20–23	BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error. Note that BNK is invalid if the error was not caused by parity checks.
24–30	—	Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid. 0 Captured error attributes and address are not valid. 1 Captured error attributes and address are valid.

10.3.1.14 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in Figure 10-17.

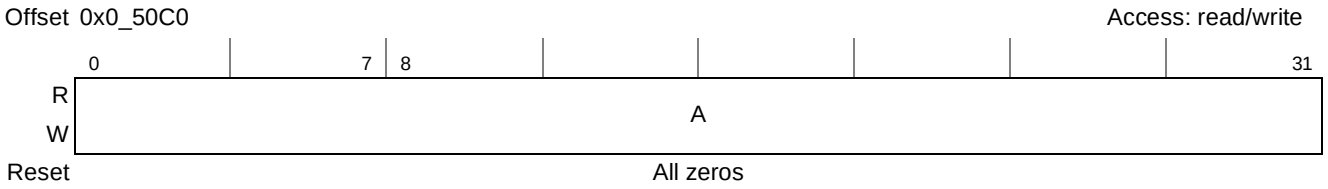


Figure 10-17. Transfer Error Address Register (LTEAR)

Table 10-20 describes LTEAR[A].

Table 10-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Holds the 32-bit address of the transaction resulting in an error.

10.3.1.15 Local Bus Configuration Register (LBCR)

LBCR is shown in Figure 10-18.

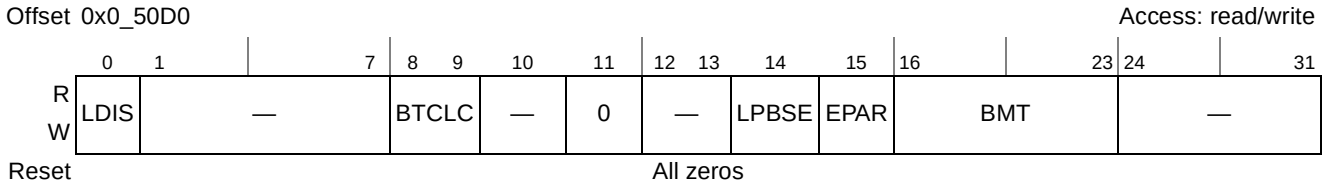


Figure 10-18. Local Bus Configuration Register

Table 10-21 describes LBCR fields.

Table 10-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled. 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/$\overline{R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ for GPCM accesses only. 11 Reserved.
10–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is available for memory control as LGPL4 (output) or $\overline{LGTA}$/LUPWAIT (input). 1 Parity byte select is enabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE signal is dedicated as the parity byte select output, and $\overline{LGTA}$/LUPWAIT is disabled.

Table 10-21. LBCR Field Descriptions (continued)

Bits	Name	Description
15	EPAR	Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity 1 Even parity
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 2048 bus clock cycles. For non-zero values of BMT, the number of LCLK clock cycles to count down before a time-out error is generated is given by bus cycles = BMT x 8. Apart from BMT = 0x00, the minimum value of BMT is 5, corresponding with 40 bus cycles. Shorter time-outs may result in spurious errors during SDRAM operation. See Section 10.4.1.6, “Parity Generation and Checking (LDP),”
24–31	—	Reserved

10.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register, shown inFigure 10-19, sets the system clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

NOTE

For proper operation of the system, this register setting must not be altered while local bus memories or devices are being accessed. Special care needs to be taken when running instructions from an local bus controller memory.

Offset 0x0_50D4 Access: read/write

	0	1	2	4	5	6	7	8	13	14	15	16	27	28	31
R W	DBYP	—	BUFCMDC	—		ECL		—		EADC		—		CLKDIV	
Reset	1	0	0 0	0	0	0	0	0 0 0 0	0 0 0 0			0 0 0 0	0 0 0 0	0 0 0 0	1 0 0 0

Figure 10-19. Clock Ratio Register (LCRR)

Table 10-22 describes LCRR fields.

Table 10-22. LCRR Field Descriptions

Bits	Name	Description
0	DBYP	DLL bypass. Should be set when using low bus clock frequencies if the DLL is unable to lock. When in DLL bypass mode, incoming data is captured in the middle of the bus clock cycle. 0 The DLL is enabled. 1 The DLL is bypassed (default).
1	—	Reserved
2–3	BUFCMDC	Additional delay cycles for SDRAM control signals. Defines the number of cycles to be added for each SDRAM command when LSDMR[BUFCMD] = 1. 00 4 01 1 10 2 11 3

Table 10-22. LCRR Field Descriptions (continued)

Bits	Name	Description
4–5	—	Reserved
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved
14–15	EADC	External address delay cycles.Defines the number of cycles for the assertion of LALE. 00 4 01 1 10 2 11 3
16–27	—	Reserved
28–31	CLKDIV	System (input) clock divider. Sets the frequency ratio between the (input) system clock and the memory bus clock. Only the values shown in the table below are allowed. 0000–0001 reserved 0010 2 0011 reserved 0100 4 0101–0111 reserved 1000 8 1001–1111 reserved

10.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to SDRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal DLL for bus clock generation ensures improved data set-up margins for board designs.
- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- The UPM supports refresh timers, address multiplexing of the external bus and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

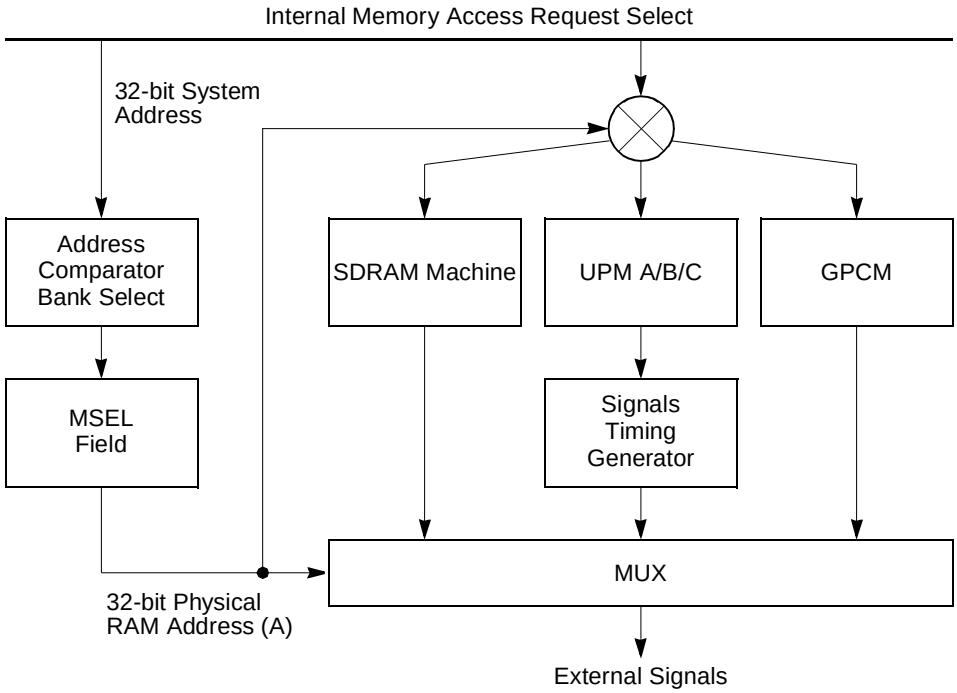


Figure 10-20. Basic Operation of Memory Controllers in the LBC

Each memory bank (chip select) can be assigned to any one of these three type of machines through the machine select bits of the base register for that bank (BR*n*[MSEL]), as illustrated in Figure 10-20. If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

10.4.1 Basic Architecture

These subsections describe the basic architecture of the LBC.

10.4.1.1 Address and Address Space Checking

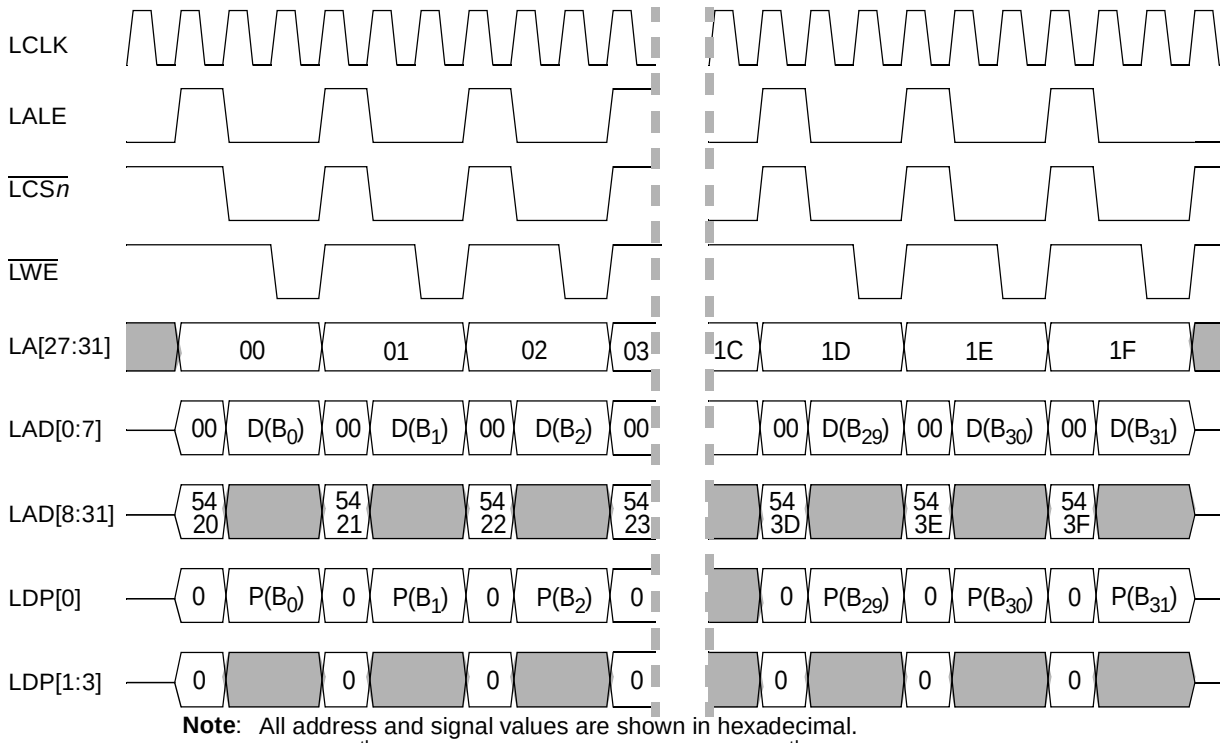
The defined base addresses are written to the BR*n* registers; the corresponding address masks are written to the OR*n* registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 17 msbs of the address, masked by OR*n*[AM], with the base address for each bank (BR*n*[BA]). If a match is found on a memory controller bank, the attributes defined in the BR*n* and OR*n* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

10.4.1.2 External Address Latch Enable Signal (LALE)

The local bus uses a multiplexed address/data bus. Therefore the LBC must distinguish between address and data phases, which take place on the same bus (LAD[0:31] signals). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address on the LAD[0:31] signals. An external address latch uses this signal to capture the address and provide it to the address signals of the memory or peripheral device. When LALE is negated, LAD[0:31] then serves as the (bi-directional) data bus for the access. Any address phase initiates the assertion of LALE, which has a programmable duration of between 1 and 4 bus clock cycles.

The frequency of LALE assertion varies across the three memory controllers. For GPCM, every assertion of $\overline{\text{LCS}n}$ is considered an independent access, and accordingly, LALE asserts prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and $\overline{\text{LCS}n}$ 32 times in order to satisfy a 32-byte cache line transfer.The SDRAM controller asserts LALE only to initiate a burst transfer with a starting address, therefore no more than one assertion of LALE may be required for SDRAM to transfer a 32-byte cache line via a 32-bit port. In the case of UPM, the frequency of LALE assertion depends on how the UPM RAM is programmed. UPM single accesses typically assert LALE once, upon commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA[27:31] with and without LALE being involved. In general, when using the GPCM and SDRAM controllers it is not necessary to use LA[27:31] if a sufficiently wide latch is used to capture the entire address during LALE phases. UPM may require LA[27:31] if the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 10-21 shows LBC signals for GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP0 are driven with valid data and parity, respectively.



Note: All address and signal values are shown in hexadecimal.
D(Bk) = kth of 32 data bytes, P(Bk) = parity bit of kth data byte.

Figure 10-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

10.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which the LBC asserts TA internally. In LBC debug mode, TA is also visible externally on the LDVAL signal. GPCM and SDRAM controllers automatically generate TA according to the timing parameters programmed for them in option and mode registers; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set. Figure 10-22 shows LALE, TA (internal), and $\overline{\text{LCS}n}$. 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.

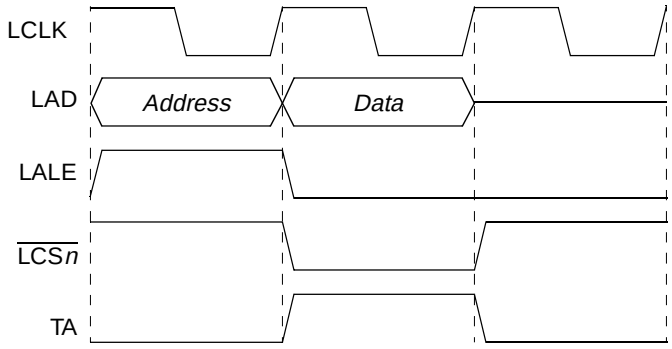


Figure 10-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}}n$

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]. Access to an SDRAM machine controlled bank does not activate the LBCTL control. LBCTL can be further configured by LBCR[BCTLC] to act as an extra $\overline{\text{LWE}}$ or an extra $\overline{\text{LOE}}$ signal when in GPCM mode.

If LBCTL is configured as a data buffer control (LBCR[BCTLC] = 00), the signal is asserted (high) on the rising edge of the bus clock on the first cycle of the memory controller operation, coincident with LALE. If the access is a write, LBCTL remains high for the whole duration. However, if the access is a read, LBCTL is negated (low) with the negation of LALE so that the memory device is able to drive the bus. If back-to-back read accesses are pending, LBCTL is asserted (high) one bus clock cycle before the next transaction starts (that is, one bus clock cycle before LALE) to allow a whole bus cycle for the bus to turn around before the next address is driven.

If an external bus transceiver is used, LBCTL should be used to signify the write direction when high. Note that the default (reset and bus idle) value of LBCTL is also high.

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 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming BR*n*[DECC]. Parity is generated and checked on a per-byte basis using LDP[0:3] for the bank if BR*n*[DECC] = 01 (normal parity) or BR*n*[DECC] = 10 for read-modify-write (RMW) parity. Byte lane parity on LDP[0:3] is generated regardless of the BR*n*[DECC] setting. Note that RMW parity can be used only for 32-bit port size banks. LBCR[EPAR] determines the global type of parity (odd or even).

10.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor starts counting down from the time-out value (LBCR[BMT]) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the countdown until the data tenure completes and then idles if there is no pending transaction.Setting LTEDR[BMD] disables bus monitor error checking (i.e. the LTESR[BM] bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in Section 10.4.4.1.4, “Exception Requests,”) or terminate a GPCM access.

It is very important to ensure that the value of LBCR[BMT] is not set too low; otherwise spurious bus time-outs may occur during normal operation—particularly for SDRAMs— resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles), LBCR[BMT] must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

10.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BR*n* and OR*n*.

Figure 10-23 shows a simple connection between an 8-bit port size SRAM device and the LBC in GPCM mode. Byte-write enable signals ($\overline{\text{LWE}}$) 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.

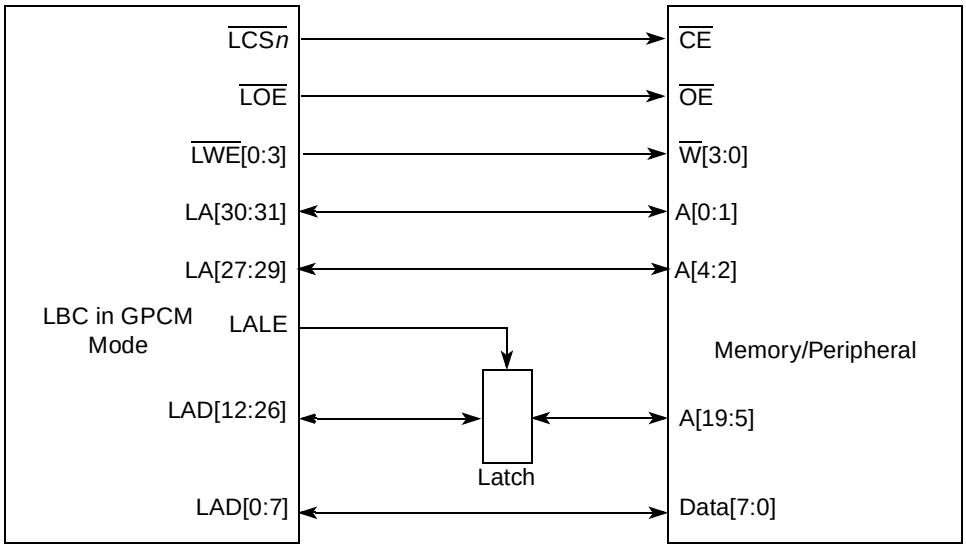


Figure 10-23. Local Bus to GPCM Device Interface

Figure 10-24 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.

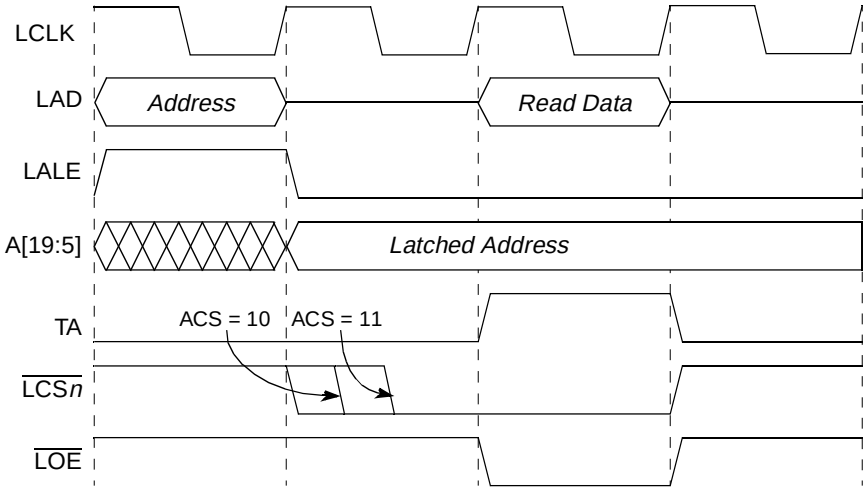


Figure 10-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

10.4.2.1 Timing Configuration

If BR*n*[MSEL] selects the GPCM, the attributes for the memory cycle are taken from OR*n*. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR and SETA fields. Table 10-23 shows signal behavior and system response for a write access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 10-24 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 10-25 and Table 10-26 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 10-23. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
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 10-24. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	Total Cycles[1]
0	0	0	00	0	1	4+SCY
0	0	0	10	1/4	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/4	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/4	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/4	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

[1] Total cycles when LALE is asserted for one cycle only (OR*n*[EAD] = 0; OR*n*[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 10-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCSn}}$ Asserted	$\overline{\text{LCS}n}$ 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 10-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ Negated to Address Change	Total Cycles[1]
0	0	0	00	0	1	4+SCY
0	0	0	10	1/2	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/2	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/2	5	9+2*SCY
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[0:31]. That is, the chip select does not assert during LALE).
- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4, 8).

- One half of a clock cycle later (for LCRR[CLKDIV] = 2, 4, or 8).
- One clock cycle later (for LCRR[CLKDIV] = 4), when OR*n*[XACS] = 1.
- Two clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1.
- Three clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1 and OR*n*[TRLX] = 1.

Figure 10-24 shows two chip-select assertion timings for the case LCRR[CLKDIV] = 4 or 8. If LCRR[CLKDIV] = 2, $\overline{\text{LCS}n}$ asserts identically for OR*n*[ACS] = 10 or 11.

10.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between 0 and 30 wait states to be added to an access by programming OR*n*[SCY] and OR*n*[TRLX]. Internal generation of transfer acknowledge is enabled if OR*n*[SETA] = 0. If $\overline{\text{LGTA}}$ is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR*n*[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}n}$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR*n*[SCY] cycles to 2*OR*n*[SCY] cycles, allowing a maximum of 30 wait states.

10.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 10-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}n}$ is connected directly to $\overline{\text{CE}}$ of the memory device. $\overline{\text{LWE}}$[0:3] are connected to the respective $\overline{\text{WE}}$[3:0] signals on the memory device where each $\overline{\text{LWE}}$[0:3] signal corresponds to a different data byte.

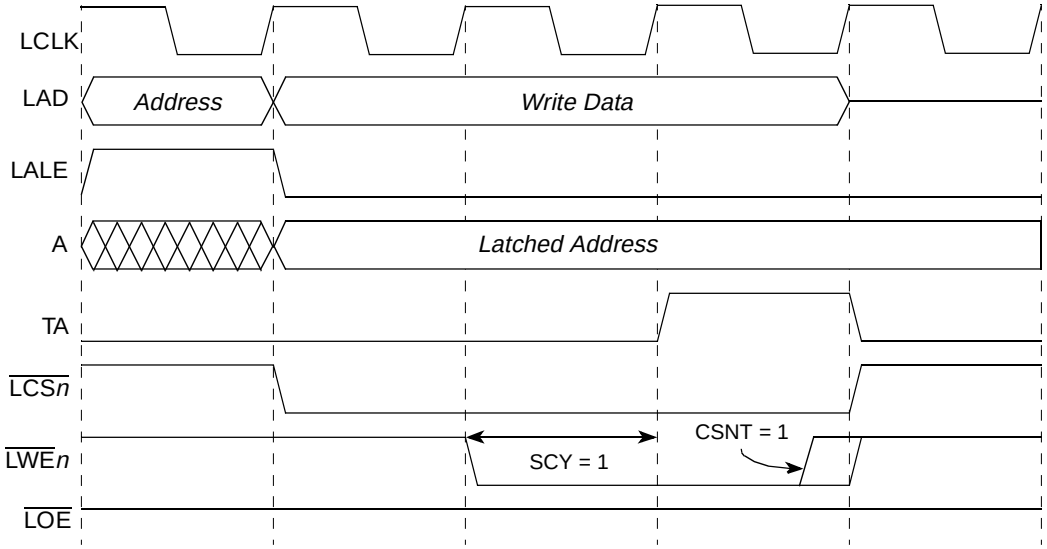


Figure 10-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)

As Figure 10-25 shows, the timing for $\overline{\text{LCS}n}$ 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 10-25. 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).
- $\overline{\text{LWE}}$[0:3] signals are negated one cycle earlier during writes.

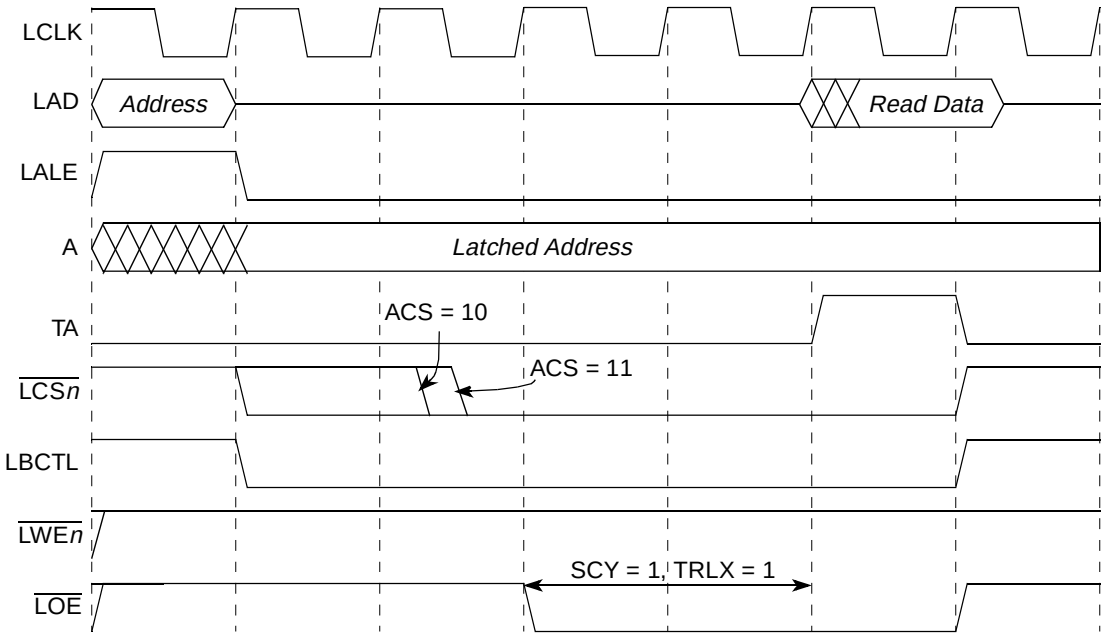


Figure 10-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

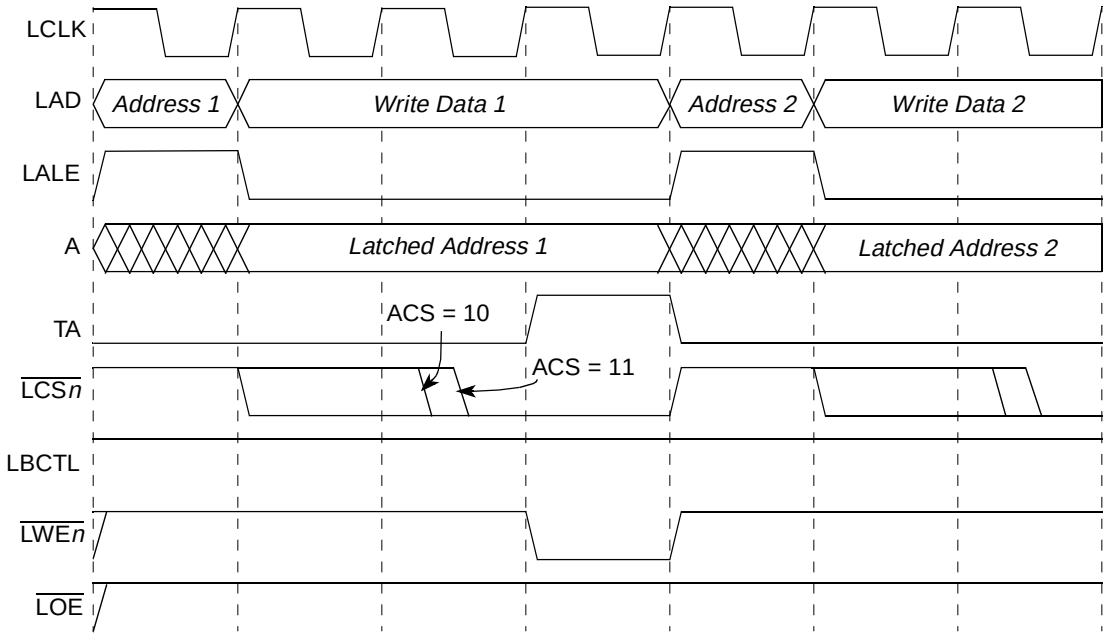


Figure 10-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

Figure 10-26 and Figure 10-27 show relaxed timing read and write transactions. The effect of CLKDIV = 2 for these examples is only to delay the assertion of $\overline{\text{LCS}n}$ in the ACS = 10 case to the ACS = 11 case. The example in Figure 10-27 also shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

When TRLX and CSNT are set in a write access, the $\overline{\text{LWE}}$[0:3] strobe signals are negated one clock earlier than in the normal case, as shown in Figure 10-28 and Figure 10-29. If ACS ≠ 00, $\overline{\text{LCS}n}$ is also negated one clock earlier.

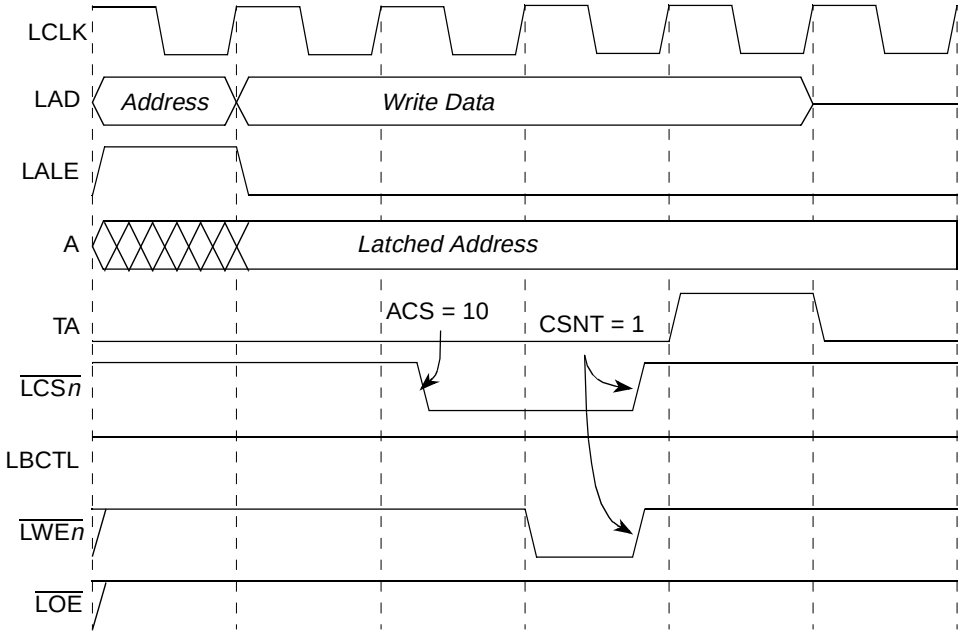


Figure 10-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

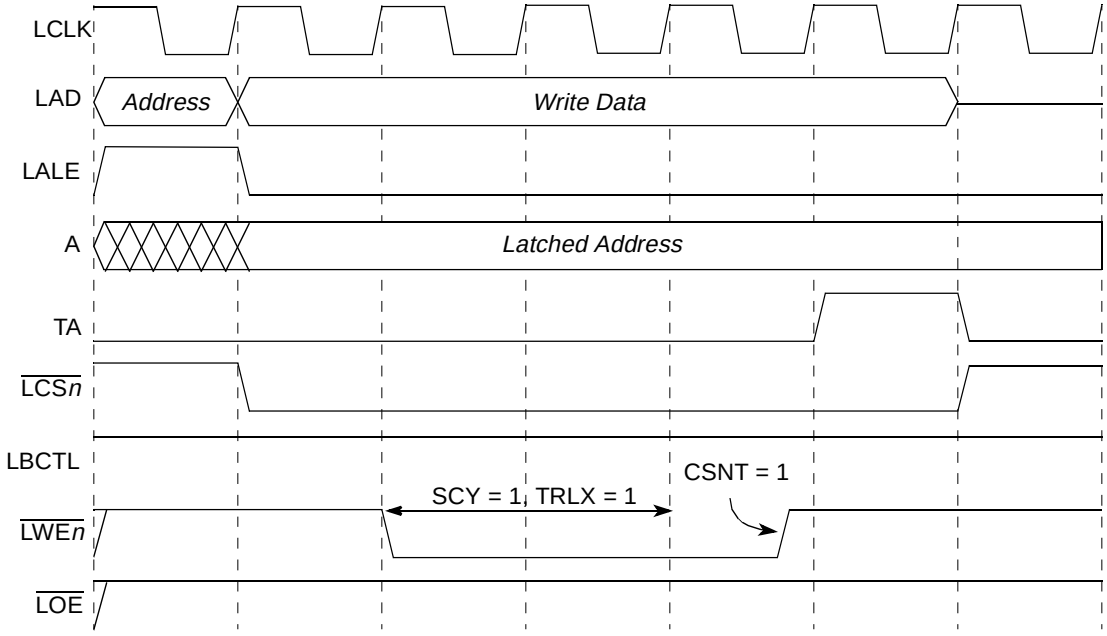


Figure 10-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

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].

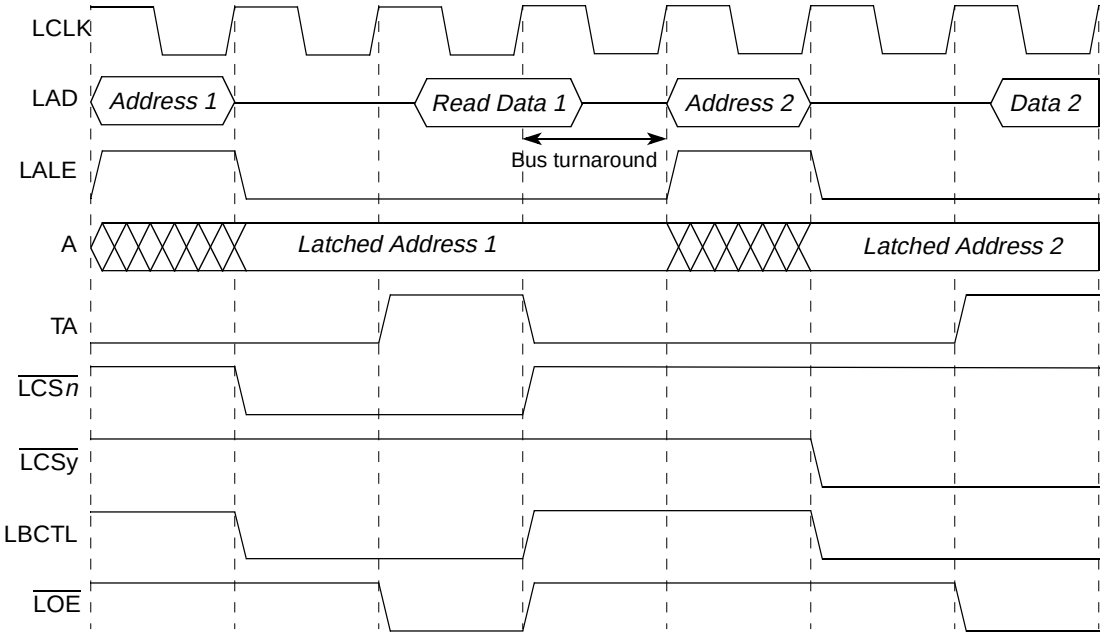


Figure 10-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

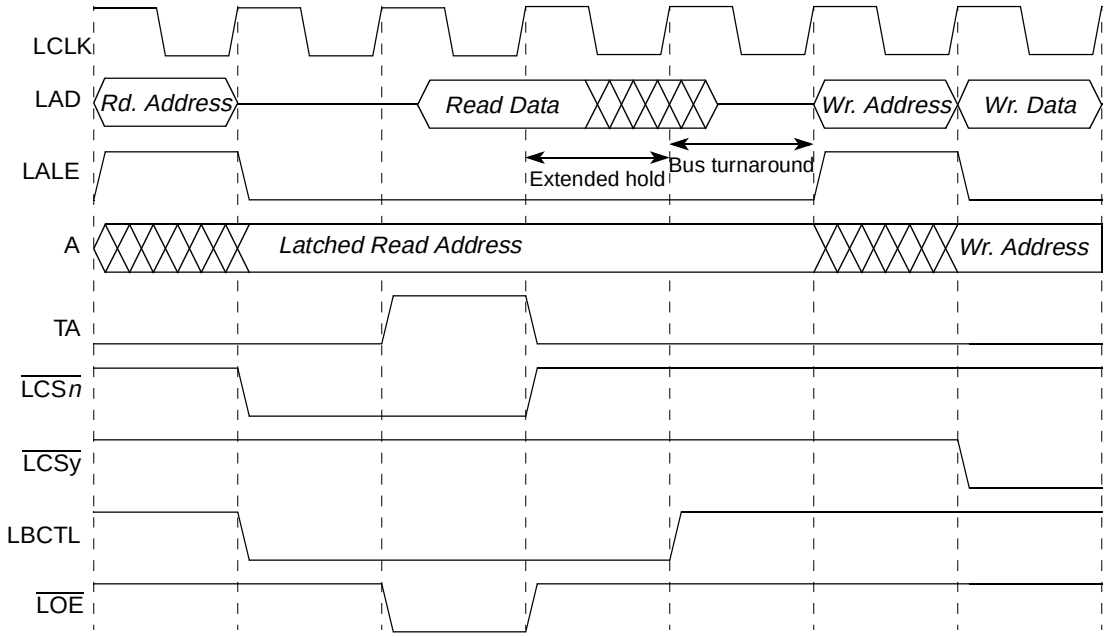


Figure 10-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)

10.4.2.3 External Access Termination ($\overline{\text{LGTA}}$)

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCS}n}$, 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 10-32.

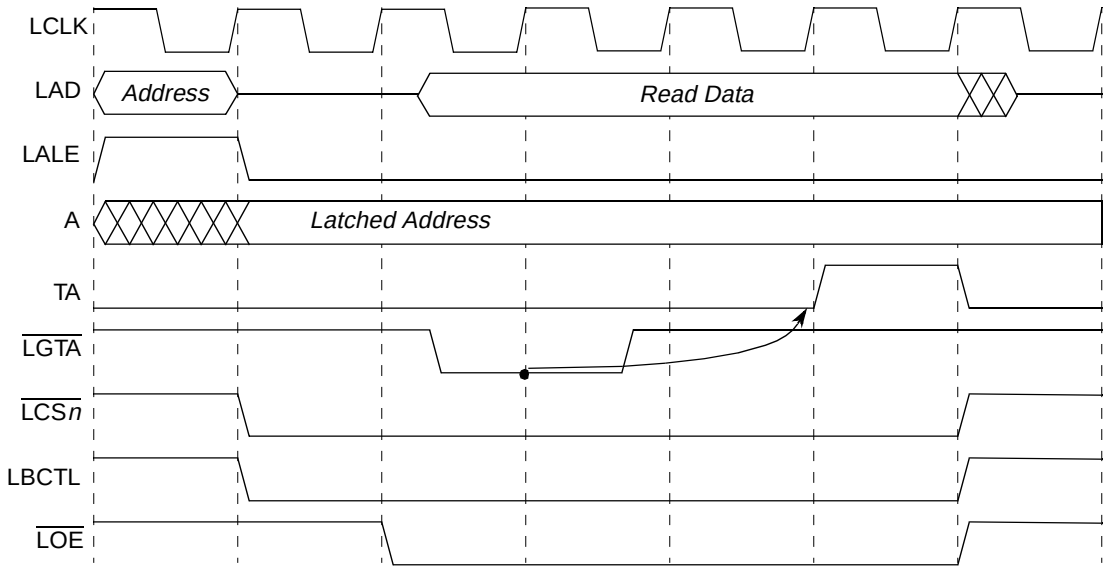


Figure 10-32. External Termination of GPCM Access

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 10-27 describes the initial values of the boot bank in the memory controller.

Table 10-27. Boot Bank Field Values After Reset

Register	Field	Setting
BR0	BA PS DECC WP MSEL ATOM V	0000_0000_0000_0000_0 From RCWH[ROMLOC] 00 0 000 00 1

Table 10-27. 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 SDRAM Machine

The LBC provides an SDRAM interface (machine) for the local bus. The machine provides the control functions and signals for Intel PC133 and JEDEC-compliant SDRAM devices. Each bank can control an SDRAM device on the local bus.

10.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR). Figure 10-33 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[2:0] of the SDRAM connect directly to LA[27:29], address signal A10 connects to the LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

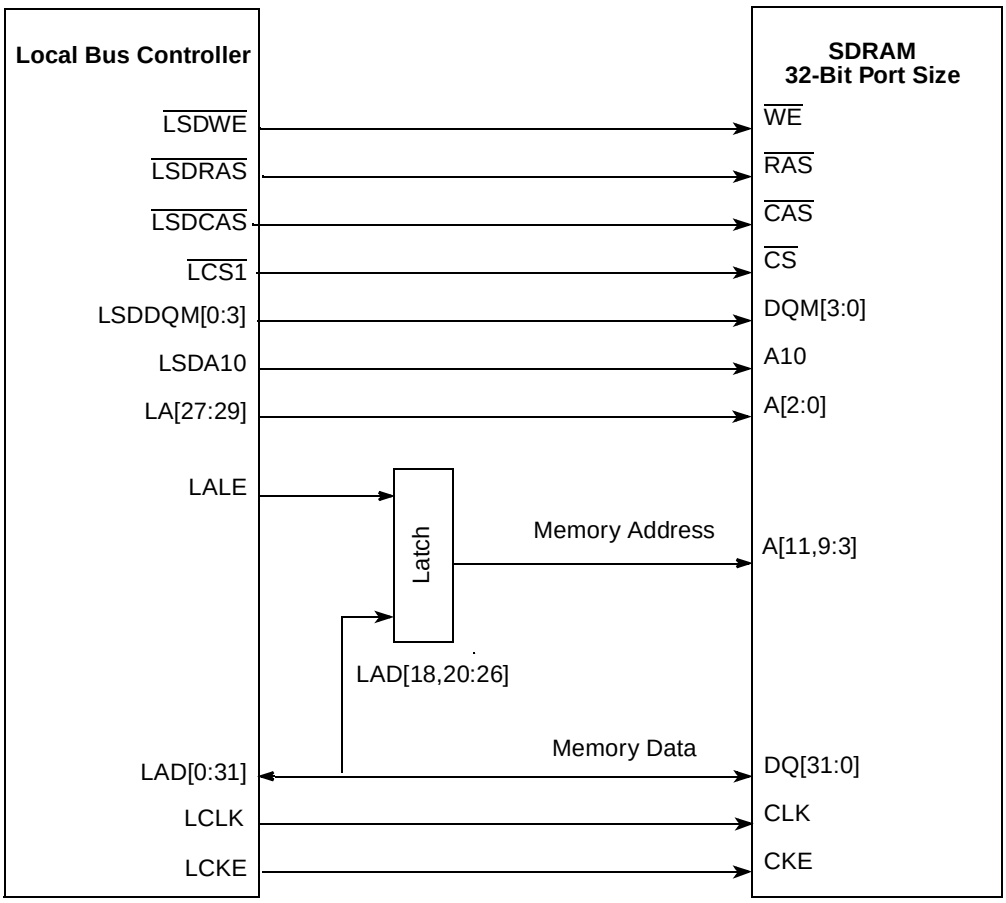


Figure 10-33. Connection to a 32-Bit SDRAM with 12 Address Lines

10.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR*n*, BR*n*, LSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

- Issue a PRECHARGE-ALL-BANKS command
- Issue eight AUTO-REFRESH commands
- Issue a MODE-SET command to initialize the mode register

The initial commands are executed by setting LSDMR[OP] and accessing the SDRAM with any write that hits the relevant bank. Since the result of any update to the LSDMR must be in effect before accessing the SDRAM with any write, a write to LSDMR should be followed immediately by a read from LSDMR, which must complete prior to an initial write to SDRAM. Further, the first write to SDRAM should be followed immediately by an SDRAM read, which must complete prior to additional LSDMR updates. This enforces a proper ordering between updates to the LSDMR and write accesses to the SDRAM. If the initialization is being done by the e300, this described protocol is guaranteed only if the SDRAM is mapped as cache-inhibited and guarded, as the CCSR memory region containing LSDMR should be. If the

initialization is from an external host, said host must ensure completion of LSDMR and SDRAM reads prior to subsequent writes, as described above.

Note that software should ensure that no memory operations begin until this process completes.

NOTE

In general (not only during power-on reset) the LSDMR/SDRAM access ordering protocol should be observed for proper operation.

10.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands

The SDRAM machine performs all accesses to SDRAM by using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

The following SDRAM interface commands are provided by setting LSDMR[OP] to a non-zero value (LSDMR[OP] = 000 sets normal read/write operation):

Table 10-28. SDRAM Interface Commands

Command (LSDMR[OP])	Description
ACTIVATE (110)	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another ACTIVATE is issued.
MODE-SET (011)	Allows setting of SDRAM options—CAS latency and burst length. CAS latency depends on the SDRAM device used. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the local bus memory controller supports only 8-beat bursts for 8-bit and 32-bit port size, or 4-beat bursts for 16-bit port size. The LBC does not support burst lengths of 1, 2 and a page for SDRAMs. The mode register data (CAS latency and burst length) is programmed into the LSDMR register by initialization software after reset. After the LSDMR is set, the LBC transfers the information to the SDRAM device by issuing a MODE-SET command.
PRECHARGE (100: single bank) (101: all-banks)	Restores data from the sense amplifiers to the appropriate row in the SDRAM device array. Also initializes the sense amplifiers to prepare for activating another row in the SDRAM device. Note that the LBC uses LSDA10 to distinguish between PRECHARGE-ALL-BANKS (LSDA10 is high) and PRECHARGE-SINGLE-BANK (LSDA10 is low). The SDRAMs must be compatible with this format.
READ (111)	Latches the column address and transfers data from the selected sense amplifier on the SDRAM device, to the output buffer as determined by the column address. During each successive clock, additional data is driven without additional read commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. Read data is discarded by the LBC.
WRITE (111)	Latches the column address and transfers data from the data signals to the selected sense amplifier on the SDRAM device, as determined by the column address. During each successive clock, additional data is transferred to the sense amplifiers from the data signals without additional write commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. LSDDQM[0:3] are inactive and write data is undefined.

Table 10-28. SDRAM Interface Commands (continued)

Command (LSDMR[OP])	Description
AUTO-REFRESH (001)	Causes a row to be read in all memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After being read, a row is automatically rewritten into the memory array. All banks must be in a precharged state before executing refresh.
SELF-REFRESH (010)	Allows data to be retained in the SDRAM device, even without any active clocks. When placed in this mode, the SDRAM device can issue its own refresh commands, without external clocking from the LBC and the LCKE signal from the LBC is negated. This command can be issued at any time. Normal operation can be resumed only by setting LSDMR[OP] = 000, and waiting a at least of 200 bus cycles before issuing reads or writes to the LBC.

10.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the OR*n* register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found, together with a bank match, the bus cycle is defined as a page hit. An open page is automatically closed by the SDRAM machine if the bus becomes idle, unless OR*n*[PMSEL] = 1.

10.4.3.5 Page Management

The LBC can manage at most four open pages (one page per SDRAM bank) for a single SDRAM device. After a page is opened, it remains open unless one of the following occurs:

- The next access is to a page in a different SDRAM device, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The next access is to a page in an SDRAM bank that has a different page open on it, in which case the old page is closed with a PRECHARGE-SINGLE-BANK command.
- The current SDRAM device requires refresh services, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- The bus becomes idle and OR*n*[PMSEL] = 0, in which case all open pages in the current device are closed with a PRECHARGE-ALL-BANKS command.

10.4.3.6 SDRAM Address Multiplexing

The lower address bus bits are connected to the memory device's address port with the memory controller multiplexing the row/column and the internal bank select lines. The position of the bank select lines are set according to LSDMR[BSMA]. Figure 10-34 shows how the SDRAM controller shifts the row address down to the lower output address signals during activate and shifts the bank select bits up to the address signals specified by LSDMR[BSMA], supporting page-based interleaving. The lsb of the logical row address (A*n* in Figure 10-34) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

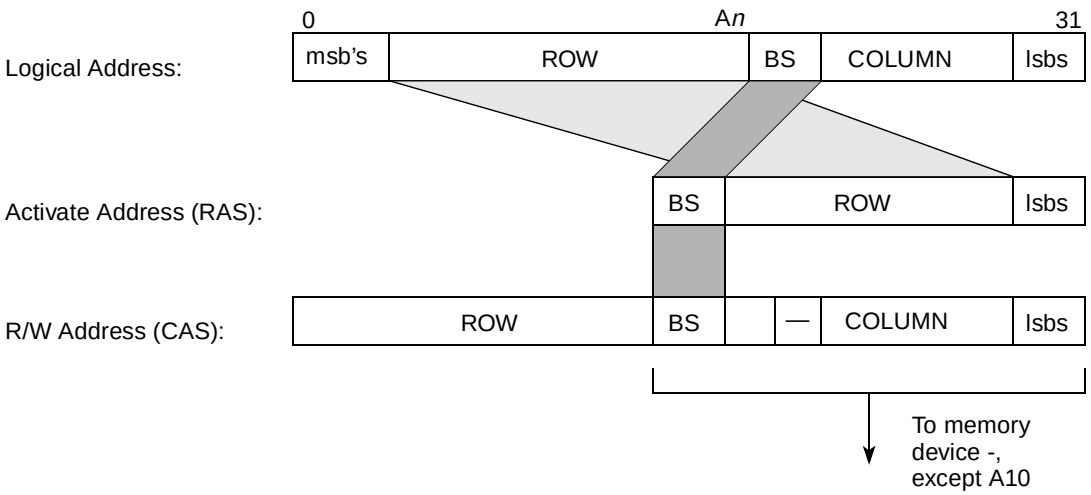


Figure 10-34. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address that includes row and column is generated on LAD[0:31]. However, address/data signal multiplexing implies that the address must be latched by an external latch that is controlled by LALE. All SDRAM device address signals need to be connected to the latched address bits and burst address bits (LA[27:31]) of the LBC, with the exception of A10, which has a dedicated connection on LSDA10. LSDA10 is driven with the appropriate row address bit for SDRAM commands that require A10 to be an address.

10.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the OR*n* and LSDMR registers. These parameters include the following:

- Precharge to activate interval (LSDMR[PRETOACT])
- Activate to read/write interval (LSDMR[ACTTORW])
- CAS latency, column address to first data out (LSDMR[CL] and LCRR[ECL])
- Write recovery, last data in to precharge (LSDMR[WRC])
- Refresh recovery interval (LSDMR[RFRC])
- External buffers on the control lines present (LSDMR[BUFCMD] and LCRR[BUFCMDC])

In addition, the LBC hardware ensures a default activate to precharge interval of 10 bus cycles. The following sections describe SDRAM parameters programmed in LSDMR.

10.4.3.7.1 Precharge-to-Activate Interval

The precharge-to-activate interval parameter, controlled by LSDMR[PRETOACT], defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

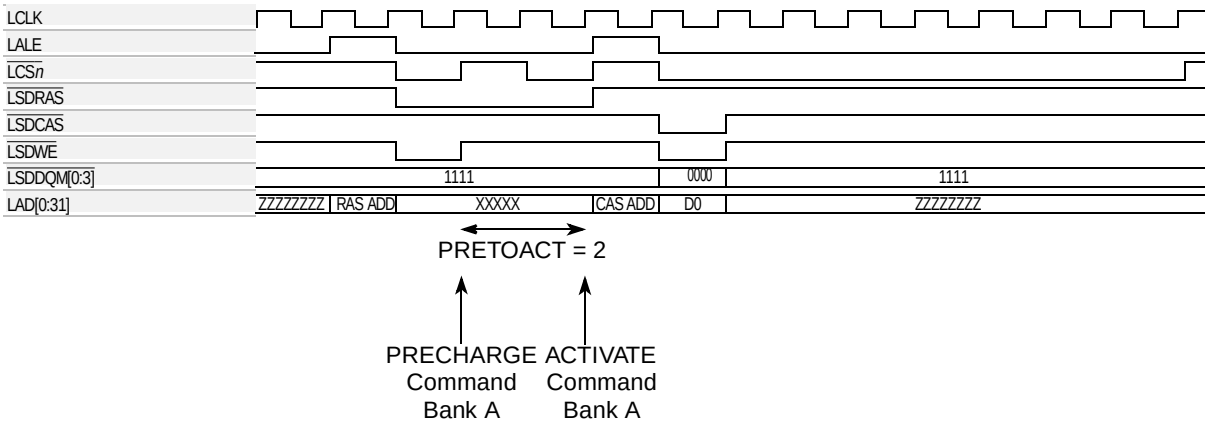


Figure 10-35. PRETOACT = 2 (2 Clock Cycles)

10.4.3.7.2 Activate-to-Read/Write Interval

This active-to-read/write interval parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

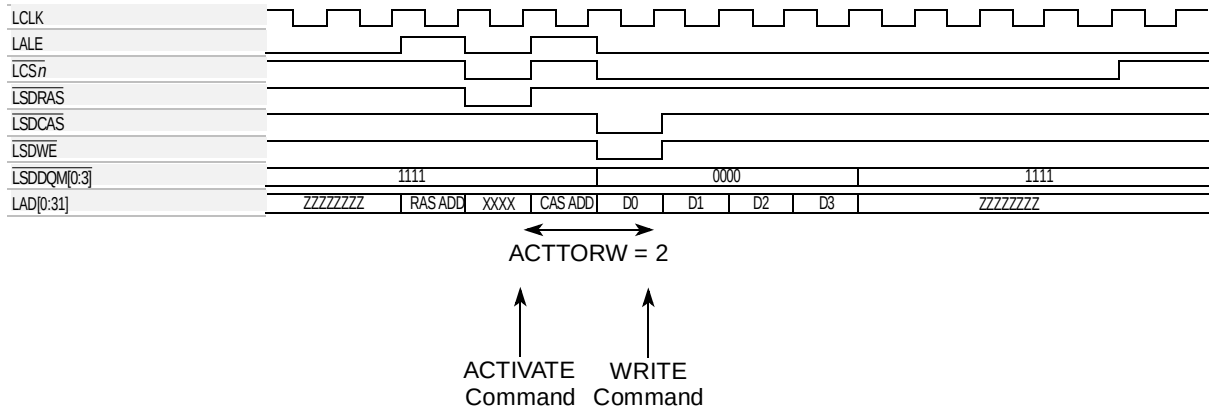


Figure 10-36. ACTTORW = 2 (2 Clock Cycles)

10.4.3.7.3 Column Address to First Data Out—CAS Latency

This parameter, controlled by LSDMR[CL] for latency of 1, 2, or 3 and by LCRR[ECL] for latency of more than 3, defines the timing for first read data after a column address is sampled by the SDRAM.

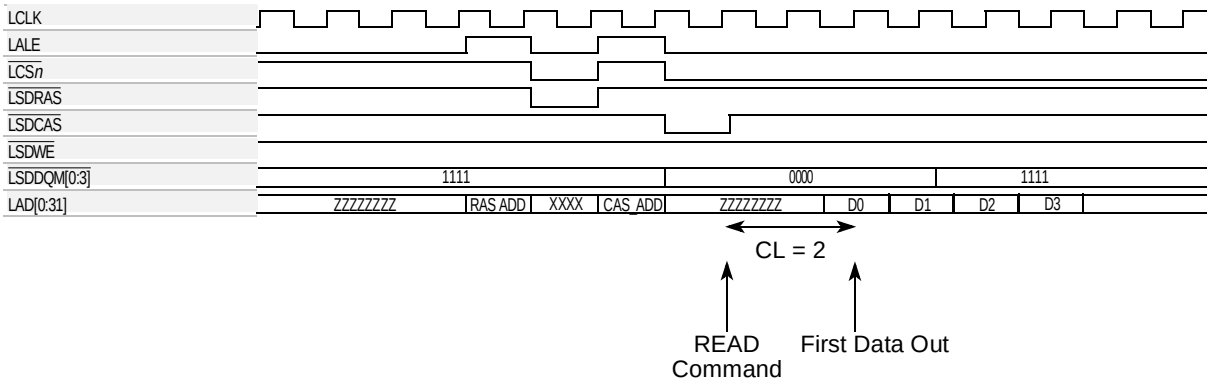


Figure 10-37. CL = 2 (2 Clock Cycles)

10.4.3.7.4 Last Data In to Precharge—Write Recovery

This parameter, controlled by LSDMR[WRC], defines the earliest timing for a PRECHARGE command after the last data was written to the SDRAM.

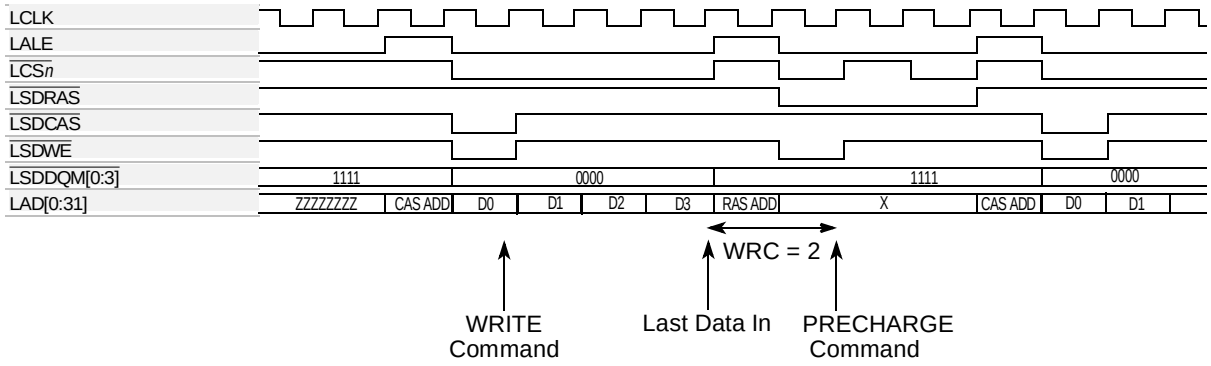


Figure 10-38. WRC = 2 (2 Clock Cycles)

10.4.3.7.5 Refresh Recovery Interval (RFRC)

This parameter, controlled by LSDMR[RFRC], defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command to the same SDRAM device.

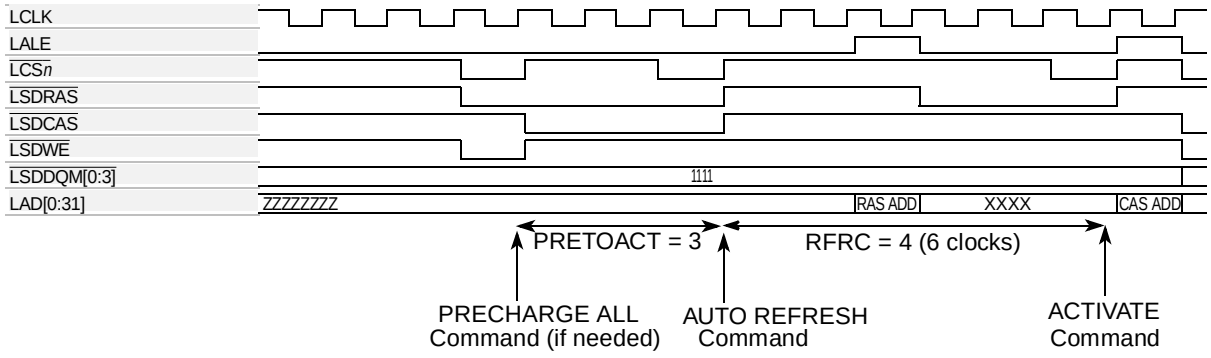


Figure 10-39. RFRC = 4 (6 Clock Cycles)

10.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and LSDA10), is endangering the device setup time, LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add LCRR[BUFCMDC] extra bus cycles to the assertion of SDRAM control signals ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and LSDA10) for each SDRAM command.

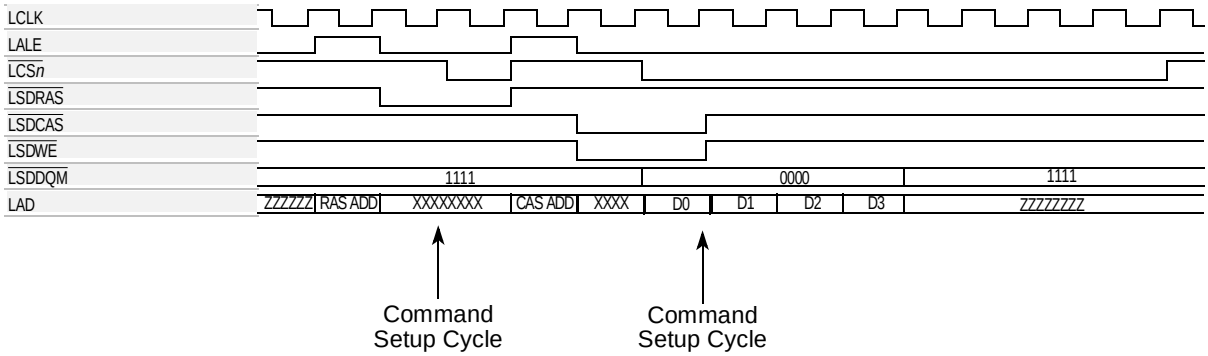


Figure 10-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

10.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

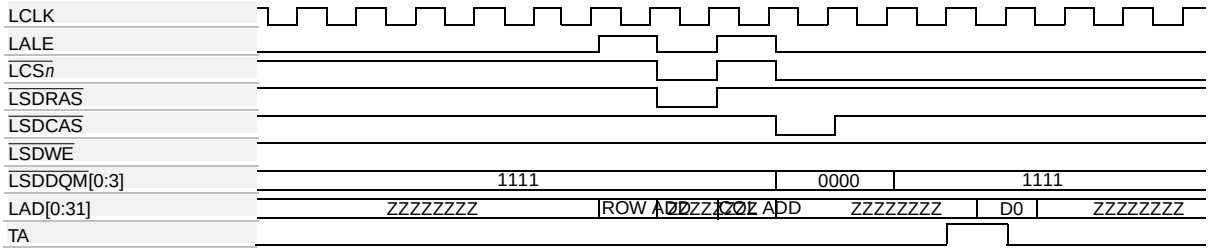


Figure 10-41. SDRAM Single-Beat Read, Page Closed, CL = 3

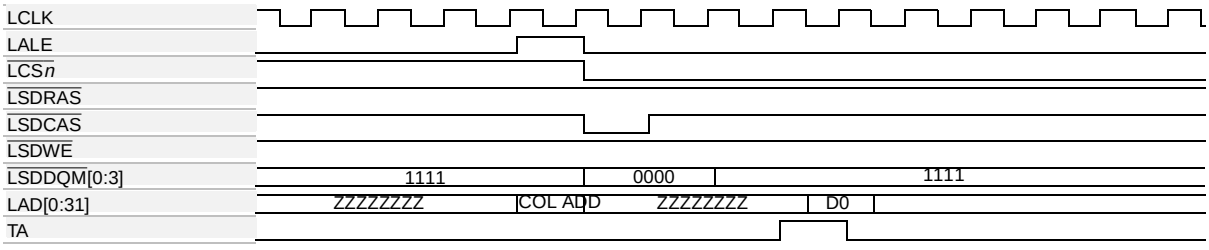


Figure 10-42. SDRAM Single-Beat Read, Page Hit, CL = 3

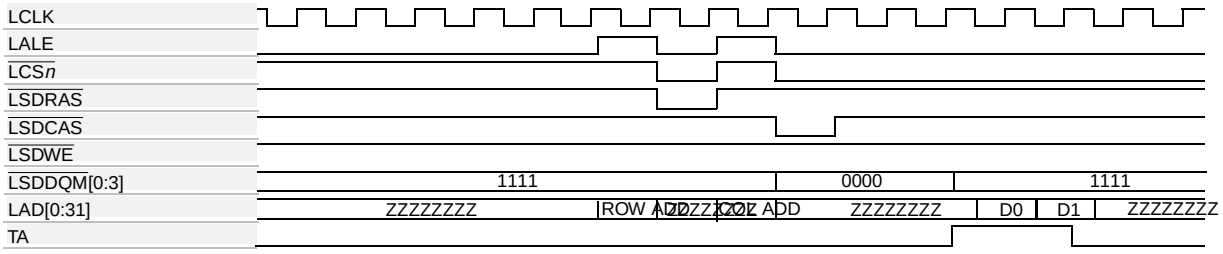


Figure 10-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

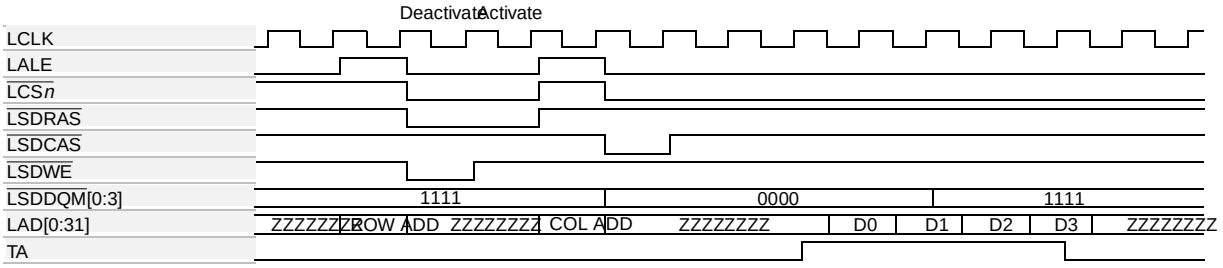


Figure 10-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

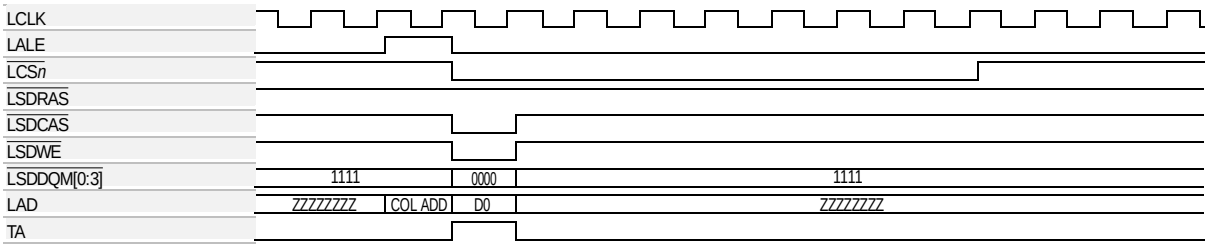


Figure 10-45. SDRAM Single-Beat Write, Page Hit

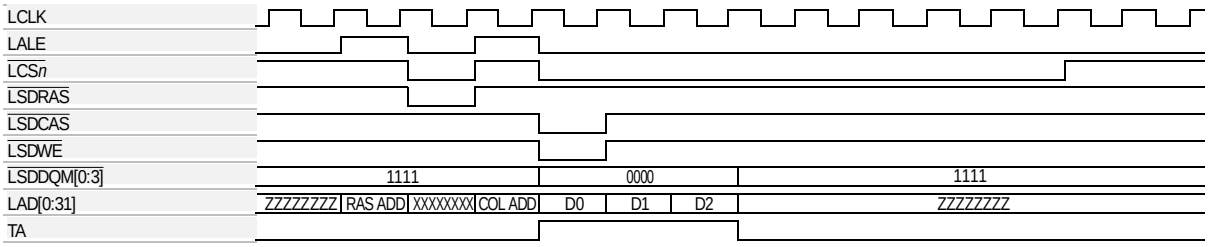


Figure 10-46. SDRAM Three-Beat Write, Page Closed

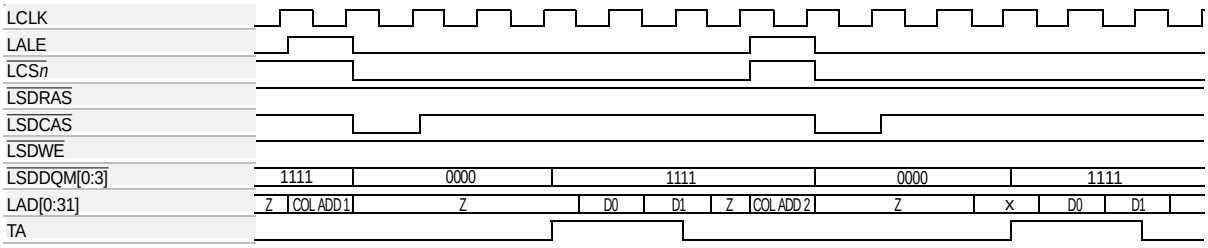


Figure 10-47. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

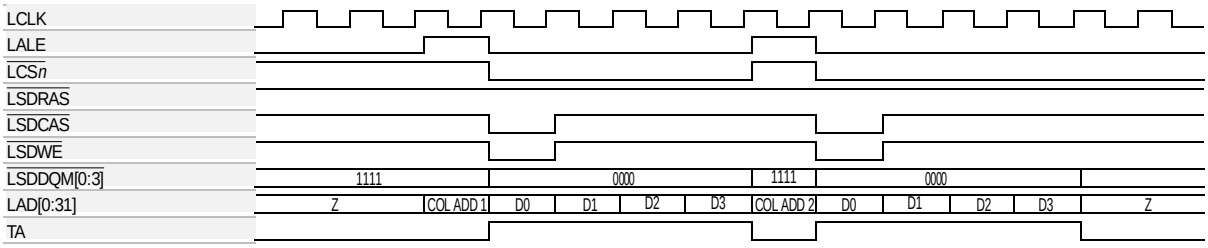


Figure 10-48. SDRAM Write-after-Write Pipelined, Page Hit

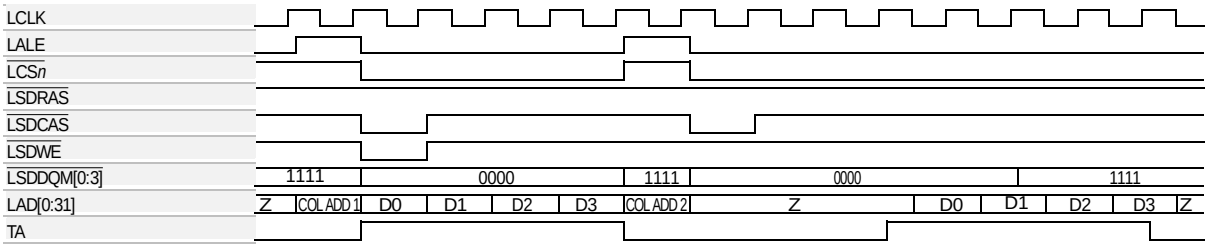


Figure 10-49. SDRAM Read-after-Write Pipelined, Page Hit

10.4.3.9 SDRAM Read/Write Transactions

The SDRAM interface supports read and write transactions of between 1 and 8 data beats for transaction sizes ranging from 1 to 32 bytes. A full burst is performed for each transaction, with the burst length dependent on the port size. A maximum burst of 8 beats is used for an 8- or 32-bit port size, while a maximum burst of 4 beats is used for a 16-bit port size, as programmed in LSDMR[BL]. For reads that require less than the full burst length, extraneous data in the burst is ignored and suppressed by the assertion of LSDDQM[0:3]. For writes that require less than the full burst length, the non-targeted addresses are protected by driving corresponding LSDDQM bits high (inactive) on the irrelevant cycles of the burst. However, system performance is not compromised because, if a new transaction is pending, the SDRAM controller begins executing it immediately, effectively terminating the burst early.

10.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 10-50. In this case, the latched address carries the mode bits for the command.

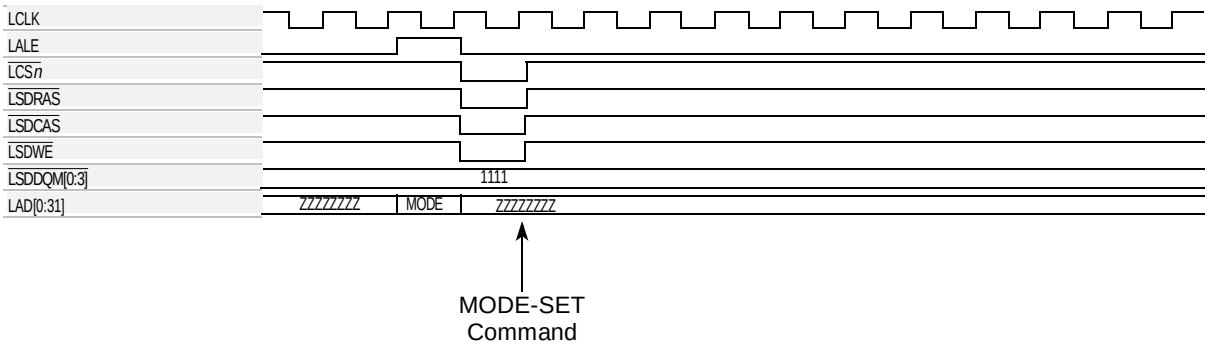


Figure 10-50. SDRAM MODE-SET Command

10.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This interval represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision between memory accesses and refresh cycles. The refresh interval period must be greater than the access time. To ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires; this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

10.4.3.11.1 SDRAM Refresh Timing

The SDRAM memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the SDRAM machine's mode register (LSDMR[RFRC]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFRC].

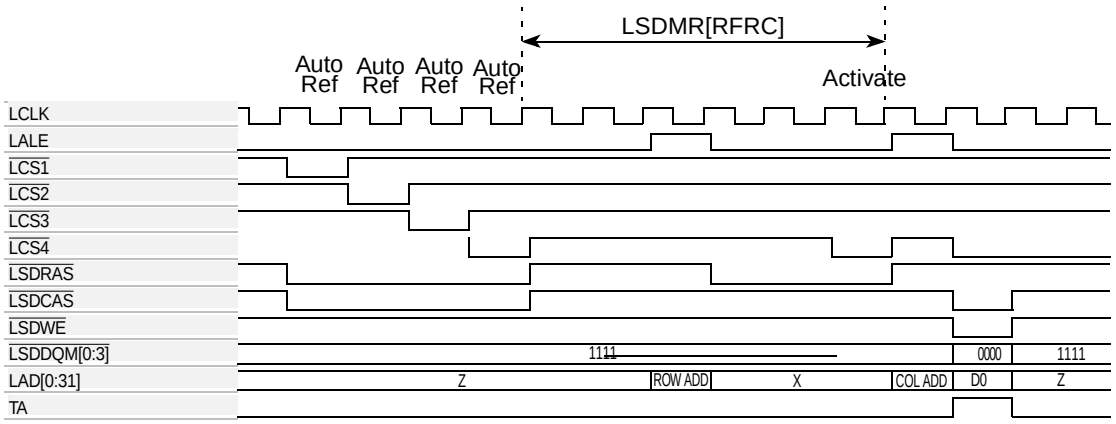


Figure 10-51. SDRAM Bank-Staggered Auto Refresh Timing

10.4.4 User-Programmable Machines (UPMs)

UPMs are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal RAM array that specifies the logical value driven on the external memory control signals ($\overline{\text{LCS}}n$, $\overline{\text{LBS}}$[0:3] 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. Figure 10-52 shows the basic operation of each UPM.

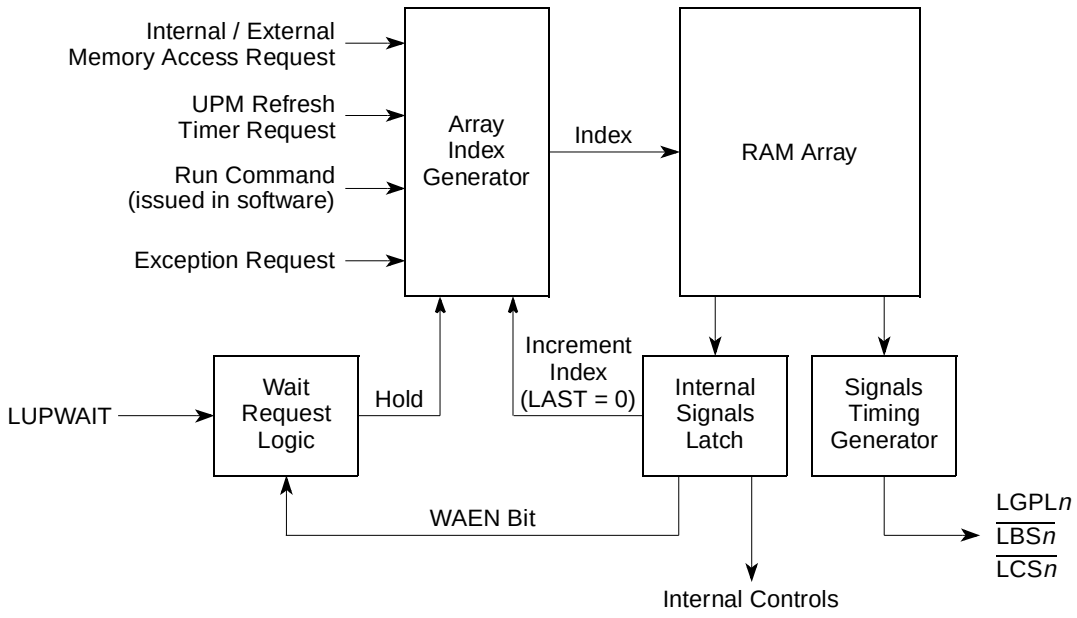


Figure 10-52. User-Programmable Machine Functional Block Diagram

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

10.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each possible UPM request. An internal device's request for a memory access initiates one of the following patterns (M*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 10-53 and Table 10-29 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.

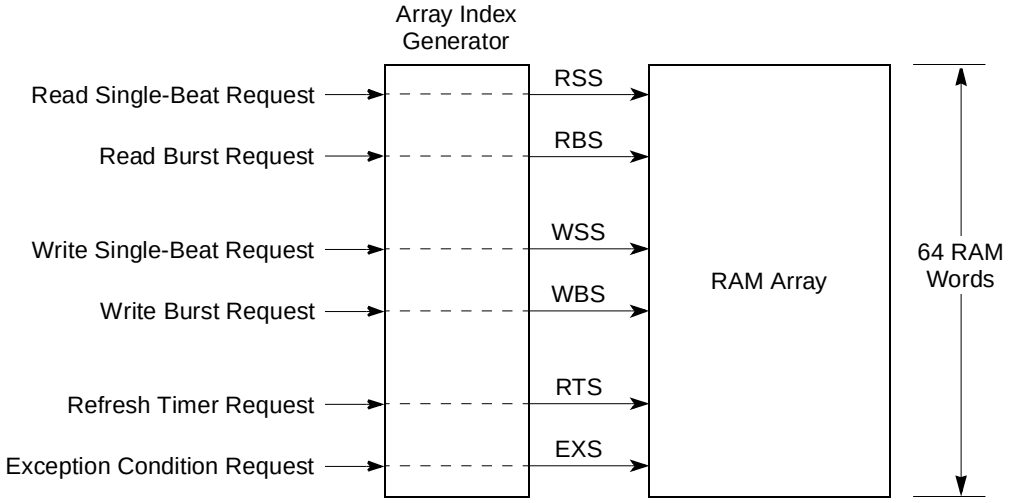


Figure 10-53. RAM Array Indexing

Table 10-29. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

10.4.4.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly four double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting OR*n*[BI]. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

10.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 10-54 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

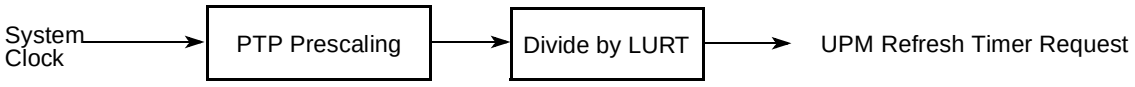


Figure 10-54. Memory Refresh Timer Request Block Diagram

By default, all local bus refreshes are performed using the refresh pattern of UPMA. This means that if refresh is required, MAMR[RFEN] must be set. It also means that only one refresh routine should be programmed and be placed in UPMA, which serves as the refresh executor. Any banks assigned to a UPM

are provided with the refresh pattern if the RFEN bit of the corresponding UPM is set. UPMA assigned banks, therefore, always receive refresh services when MAMR[RFEN] is set, while UPMB and UPMC assigned banks also receive (the same) refresh services if the corresponding M*n*MR[RFEN] bits are set.

Note that the UPM refresh timer request should not be used in a system with SDRAM refresh enabled. The system designer must choose to use either SDRAM refresh or UPM refresh. Using both may result in missing refresh periods to memory.

10.4.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The RUN command is issued by setting M*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.

10.4.4.1.4 Exception Requests

When the LBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

10.4.4.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program the UPMs:

1. Set up BR*n* and OR*n* registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT, and MAMR[RFEN] if refresh is required.
4. Program M*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 e300c1 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.

10.4.4.2.1 UPM Programming Example (Two Sequential Writes to the RAM Array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant BR*x* and OR*x* registers have been previously set up:

1. Program M*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.

10.4.4.2.2 UPM Programming Example (Two Sequential Reads from the RAM Array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (M*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.

10.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. For LCRR[CLKDIV] = 4 or 8, each bit in the RAM word relating to $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$ timing specifies the value of the corresponding external signal at each quarter phase of the bus clock. If LCRR[CLKDIV] = 2, the external signal can change value only on each half phase of the bus clock. If the RAM word in this case (LCRR[CLKDIV] = 2) specifies a quarter phase signal change, the signal timing generator interprets this as a half cycle change.

The division of UPM bus cycles into phases is shown in Figure 10-55 and Figure 10-56. If LCRR[CLKDIV] = 2, the bus cycle comprises only two active phases, T1 and T3, which correspond with the first and second halves of the bus clock cycle, respectively. However, if LCRR[CLKDIV] = 4 or 8, four phases, T1–T4, define four quarters of the bus clock cycle. Because T2 and T4 are inactive when LCRR[CLKDIV] = 2, UPM ignores signal timing programmed for assertion in either of these phases in the case LCRR[CLKDIV] = 2.

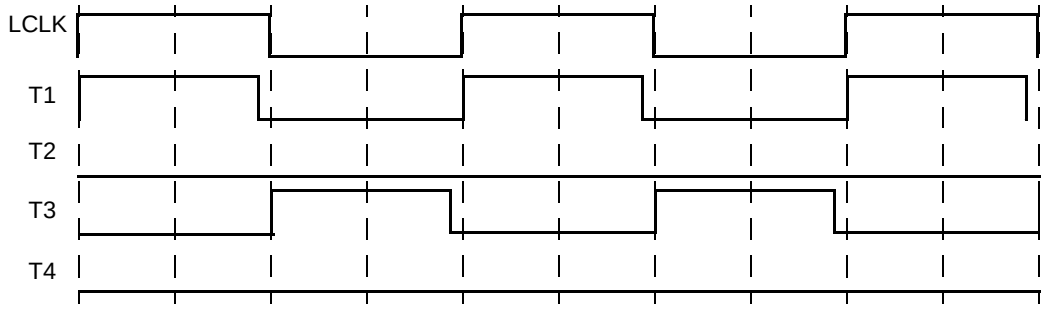


Figure 10-55. UPM Clock Scheme for LCRR[CLKDIV] = 2

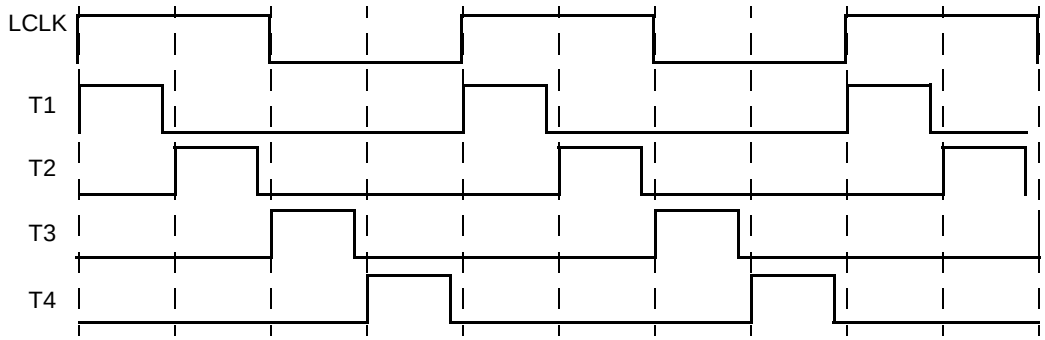


Figure 10-56. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

10.4.4.4 UPM RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 10-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}n}$ is for the bank that matches the current address. The selected $\overline{\text{LBS}}$ is for the byte lanes read or written by the access.

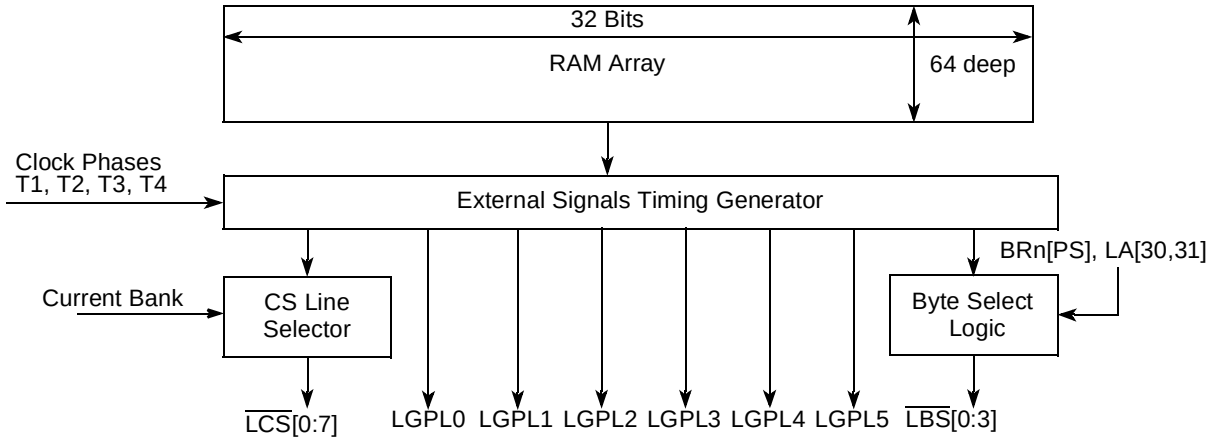


Figure 10-57. UPM RAM Array and Signal Generation

10.4.4.4.1 UPM RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 10-58 shows the RAM word fields. When LCRR[CLKDIV] = 4 or 8, the CST*n* and BST*n* bits determine the state of UPM signals $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$[0:3] at each quarter phase of the bus clock. When LCRR[CLKDIV] = 2, CST2 and CST4 are ignored and the external has the values defined by CST1 and CST3 but extended to half the clock cycle in duration. The same interpretation occurs for the BST*n* bits when LCRR[CLKDIV] = 2.

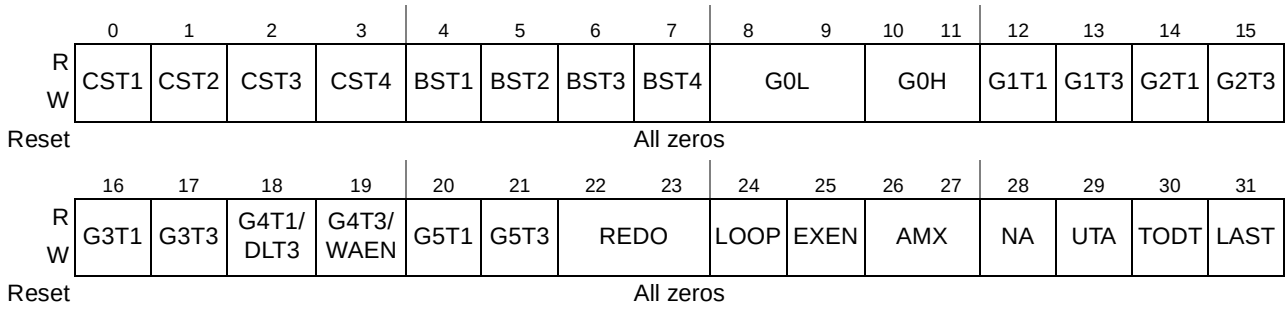


Figure 10-58. UPM RAM Word Field Descriptions

Table 10-30 describes RAM word fields.

Table 10-30. UPM RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ 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[30:31].
5	BST2	Byte select timing 2:. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 2 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR*n*[PS] and LA[30:31]. Ignored when LCRR[CLKDIV] = 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 2 (LCRR[CLKDIV] = 2), in conjunction with BR*n*[PS] and LA[30:31].
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 4 (LCRR[CLKDIV] = 4 or 8), in conjunction with BR*n*[PS] and LA[30:31]. Ignored when LCRR[CLKDIV] = 2.

Table 10-30. 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 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 10-30. 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.
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[0:31] during a LALE phase. Any change in the AMX field initiates a new LALE (address) phase. 00 LAD[0:31] is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD[0:31] is the address multiplexed according to M*n*MR[AM]. For example, row address. 11 LAD[0:31] is the contents of MAR. Used, for example, to initialize a mode. **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[27:31] is 1, 2, or 4 for port sizes of 8-bits, 16-bits and 32-bits, respectively.
29	UTA	UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle. 0 Transfer acknowledge is not asserted in the current cycle. 1 Transfer acknowledge is asserted in the current cycle.
30	TODT	Turn-on disable timer. The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a 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.

Table 10-30. UPM RAM Word Field Descriptions (continued)

Bits	Name	Description
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.

10.4.4.4.2 Chip-Select Signal Timing (CST*n*)

If BR*n*[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCS}n}$ for that bank with timing as specified in the UPM RAM word CST*n* fields. The selected UPM affects only the assertion and negation of the appropriate $\overline{\text{LCS}n}$ signal. The state of the selected $\overline{\text{LCS}n}$ signal of the corresponding bank depends on the value of each CST*n* bit. Figure 10-59 shows how UPMs control $\overline{\text{LCS}n}$ signals.

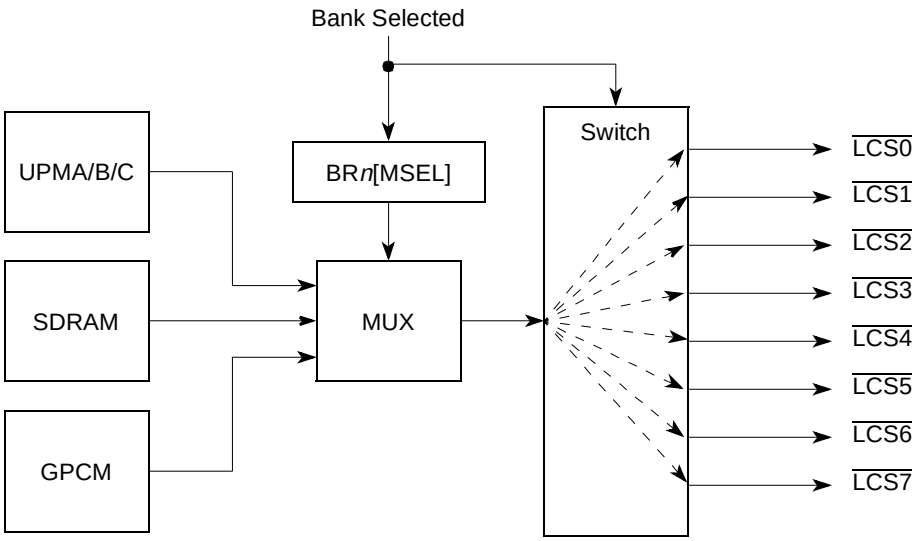


Figure 10-59. $\overline{\text{LCS}n}$ Signal Selection

10.4.4.4.3 Byte Select Signal Timing (BST*n*)

If BR*n*[MSEL] of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate $\overline{\text{LBS}}$[0:3] signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:3] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

Figure 10-60 shows how UPMs control $\overline{\text{LBS}}$[0:3].

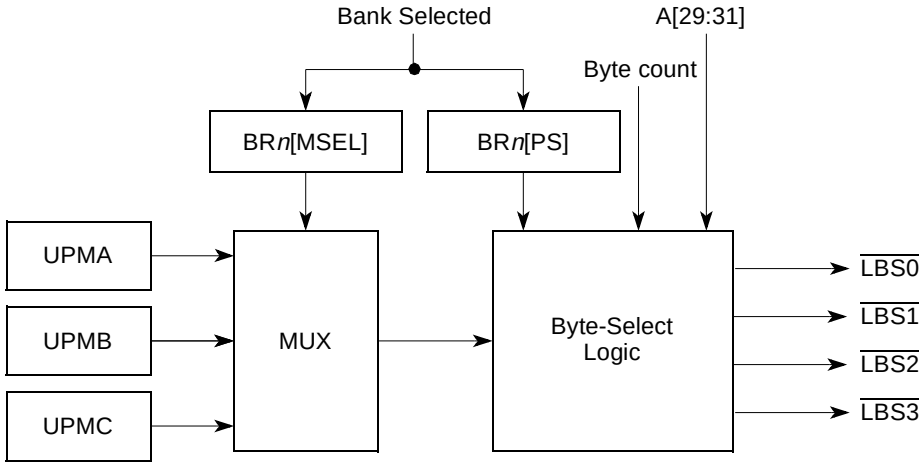


Figure 10-60. $\overline{\text{LBS}}$ Signal Selection

The uppermost byte select ($\overline{\text{LBS0}}$), when asserted, indicates that LAD[0:7] contains valid data during a cycle. Likewise, $\overline{\text{LBS1}}$ indicates that LAD[8:15] contains valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ indicates that LAD[24:31] contains valid data. For a UPM refresh timer request, all $\overline{\text{LBS}}$[0:3] signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, the $\overline{\text{LBS}}$[0:3] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

10.4.4.4.4 General-Purpose Signals (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.

10.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 10-31. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken: LAST and LOOP must not be set together.

Table 10-31. 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

10.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

10.4.4.4.7 Address Multiplexing (AMX)

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between driving the transaction address, driving it according to the multiplexing specified by the M*n*MR[AM] field, or driving the MAR contents on the address signals. In all cases, LA[27:31] of the LBC are driven by the five lsbs of the address selected by AMX, regardless of whether the NA bit of the RAM word is used to increment the current address. The effect of NA = 1 is visible only when AMX = 00 chooses the column address.

Table 10-32 shows how M*n*MR[AM] settings affect address multiplexing when the RAM word AMX = 10. The 16 msbs of the LAD[0:31] bus during an address phase are driven with zero in the AMX = 10 case.

Table 10-32. UPM Address Multiplexing

AM	LAD[0:31] as Address Signals	A0–A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external signal when address multiplexing is enabled— RAM word AMX = 10	0	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		0	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		0	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		0	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		0	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

Note that any change to AMX from one RAM word to the next RAM word executed results in an address phase on LAD[0:31] with the assertion of LALE for the number of cycles set for LALE in the OR*n* and LCRR registers. LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

10.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the LBC), the value of the DLT3 bit in the same RAM word, in conjunction with M*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 10-61 shows how data sampling is controlled by the UPM.

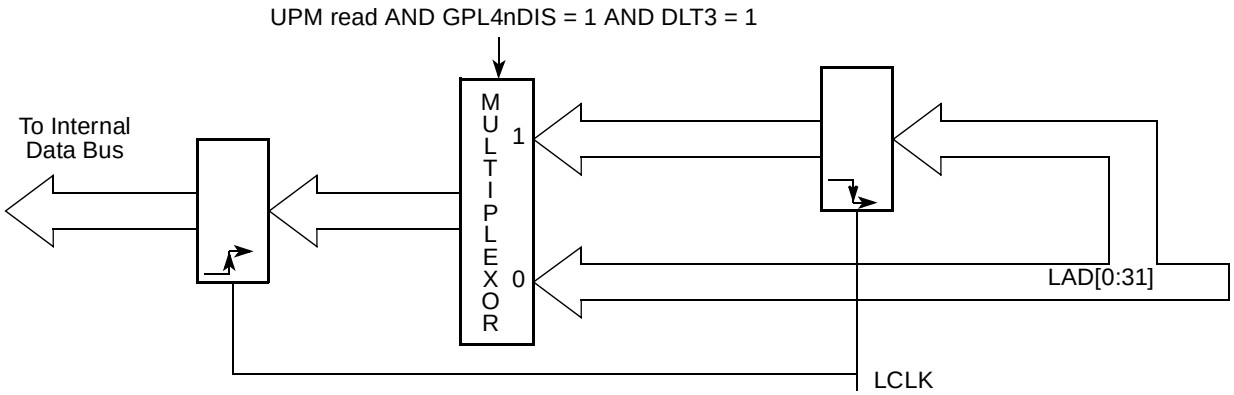


Figure 10-61. UPM Read Access Data Sampling

10.4.4.4.9 $\overline{\text{LGPL}}$[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.

10.4.4.4.10 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if LAST = 1 in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 10-62 shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the $\overline{\text{LCS}n}$ and $\overline{\text{LGPL1}}$ 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.

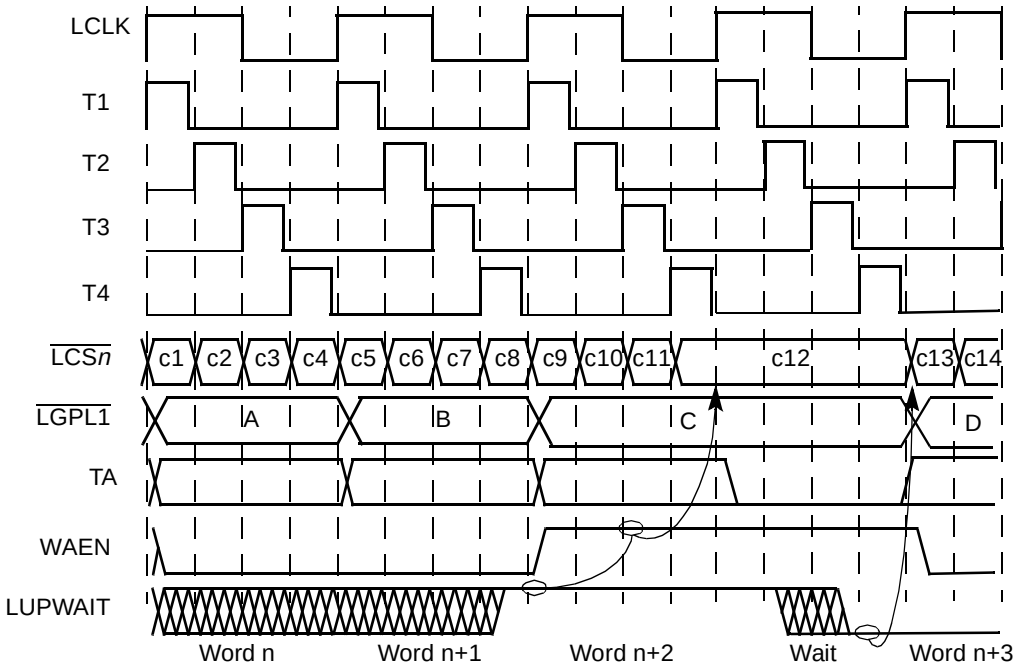


Figure 10-62. Effect of LUPWAIT Signal

10.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

However, programming WAEN = 1 and UTA = 1 in the same RAM word allows the UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether the UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to effect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

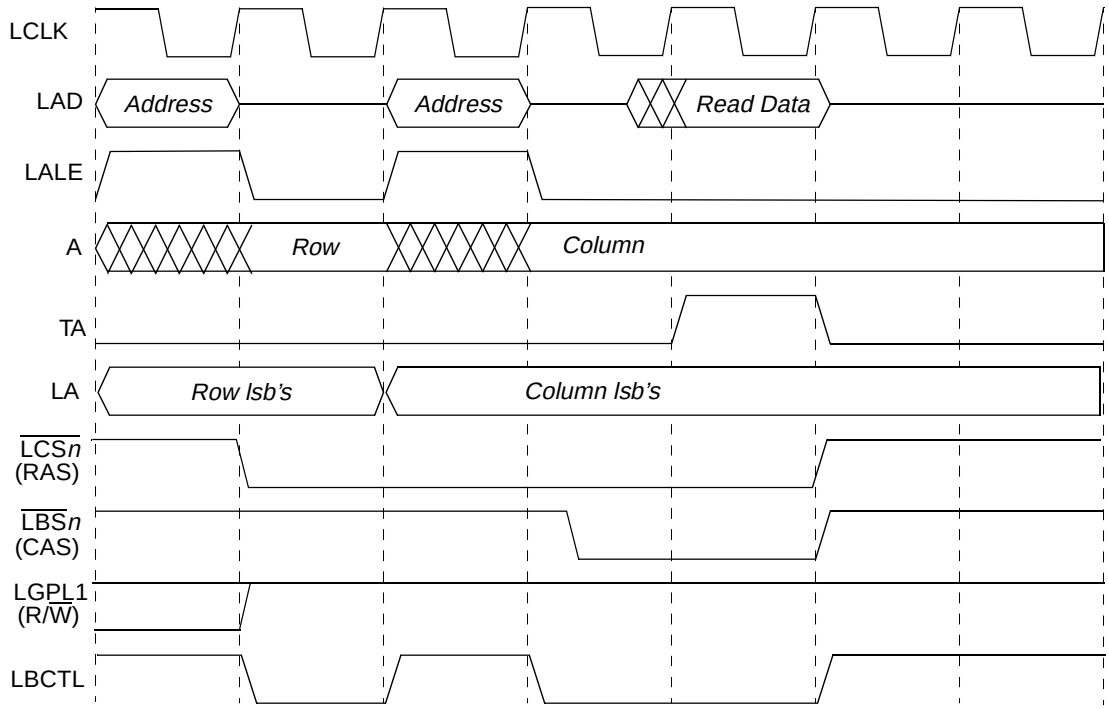
10.4.4.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of 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.

10.4.4.7 Memory System Interface Example Using UPM

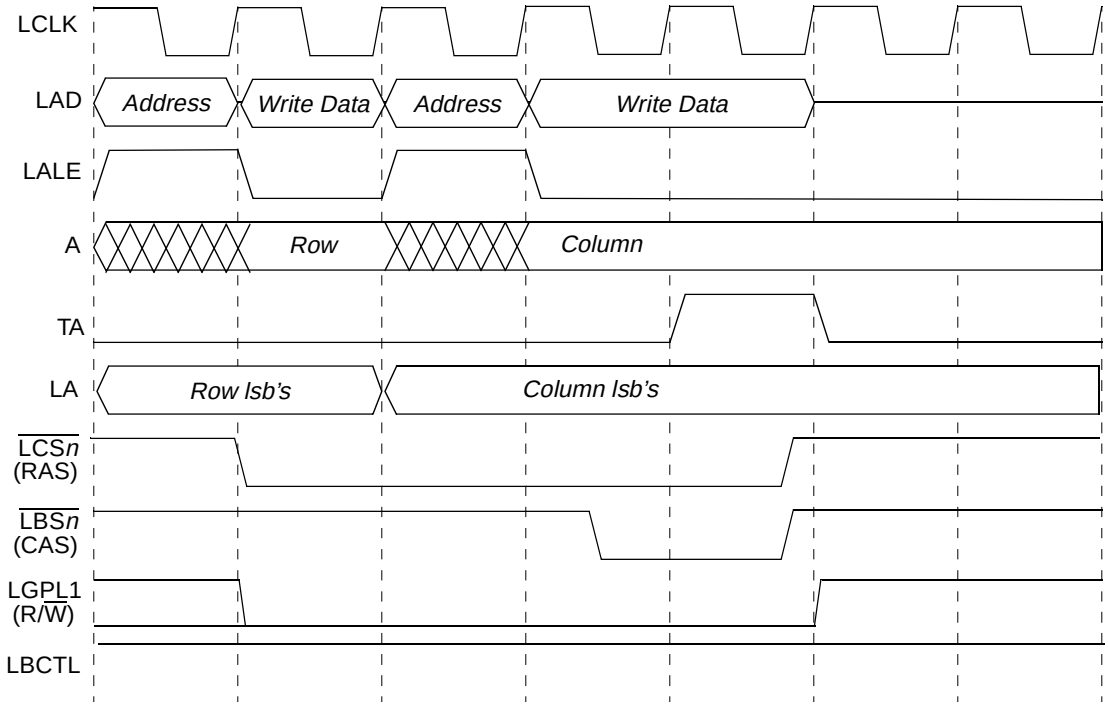
Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section presents timing diagrams for various UPM configurations, using fast-page mode DRAM as

an example, with LCRR[CLKDIV] = 4 or 8. These examples may not represent the timing necessary for any specific device used with the LBC. Here, LGPL1 is programmed to drive R/$\overline{W}$ of the DRAM, although any LGPL*n* signal can be used for this purpose.



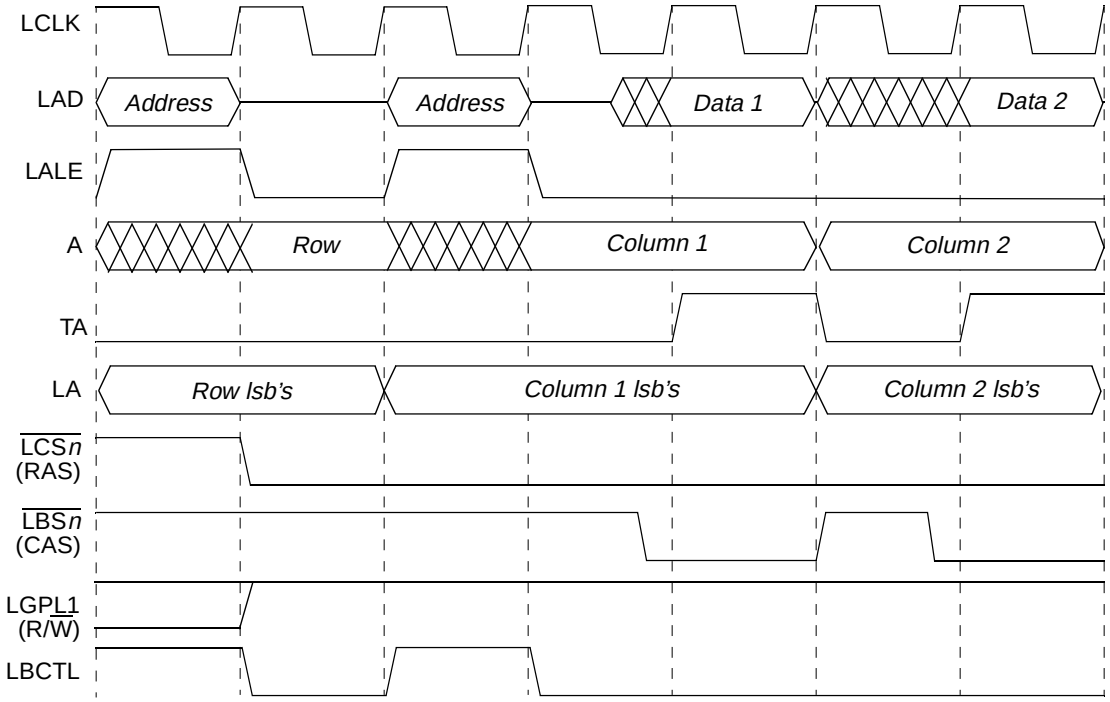
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo[0]					Bit 22
redo[1]					Bit 23
loop	0		0	0	Bit 24
exen	0		0	0	Bit 25
amx0	1		0	0	Bit 26
amx1	0		0	0	Bit 27
na	0		0	0	Bit 28
uta	0		0	1	Bit 29
todt	0		0	1	Bit 30
last	0		0	1	Bit 31
	RSS	**RSS+1**	**RSS+1**	**RSS+2**	

Figure 10-63. Single-Beat Read Access to FPM DRAM



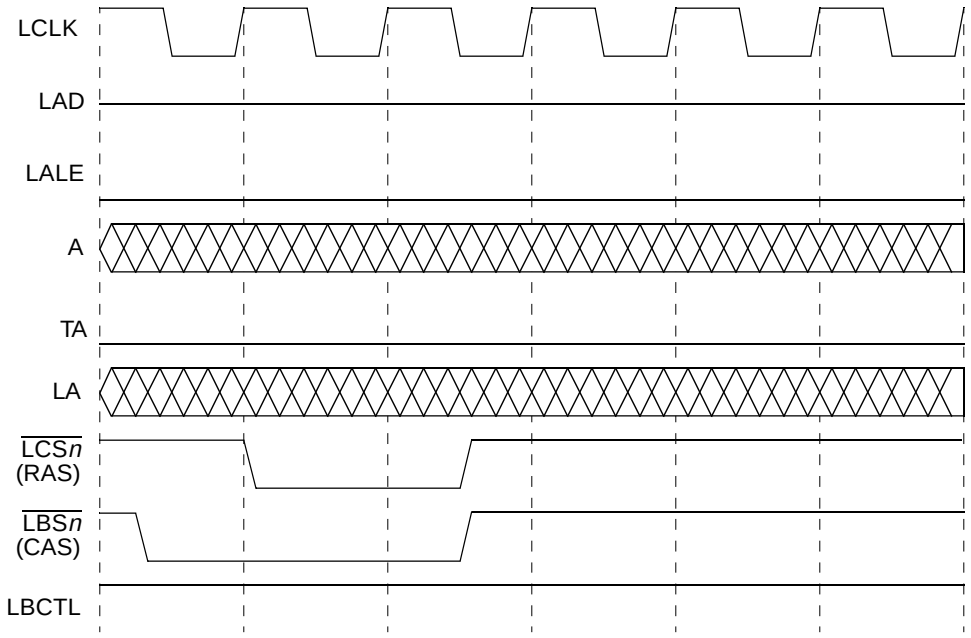
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 16 Bit 17
g4t1 g4t3					Bit 18 Bit 19
g5t1 g5t3					Bit 20 Bit 21
redo[0] redo[1]					Bit 22 Bit 23
loop exen	0 0		0 0	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	WSS	**WSS+1**	**WSS+1**	**WSS+2**	

Figure 10-64. Single-Beat Write Access to FPM DRAM



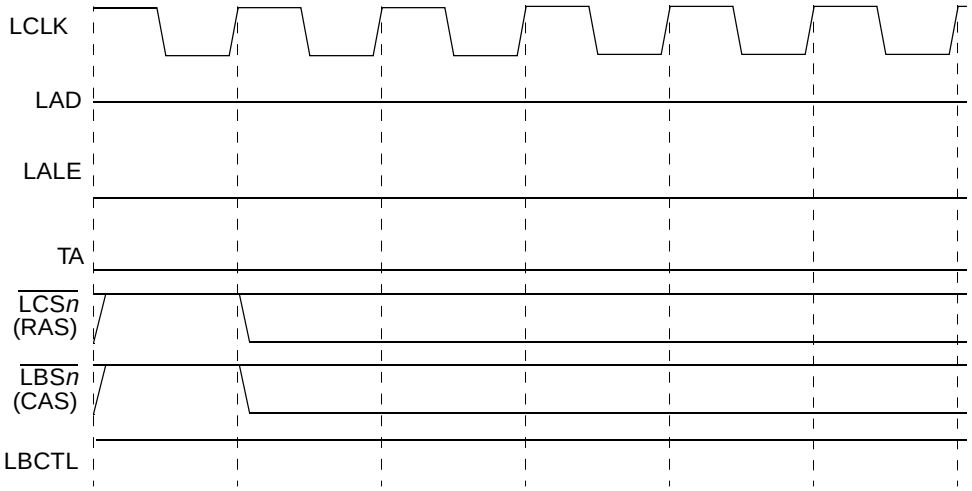
cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12
g1t3	1		1	1	1	Bit 13
g2t1						Bit 14
g2t3						Bit 15
g3t1						Bit 16
g3t3						Bit 17
g4t1						Bit 18
g4t3						Bit 19
g5t1						Bit 20
g5t3						Bit 21
redo[0]						Bit 22
redo[1]						Bit 23
loop	0		1	1	0	Bit 24
exen	0		0	1	0	Bit 25
amx0	1		0	0	0	Bit 26
amx1	0		0	0	0	Bit 27
na	0		0	1	0	Bit 28
uta	0		0	1	0	Bit 29
todt	0		0	0	1	Bit 30
last	0		0	0	1	Bit 31
	RBS		**RBS+1**	**RBS+2**	**RBS+3**	

Figure 10-65. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)



cst1	1	0	0	Bit 0
cst2	1	0	0	Bit 1
cst3	1	0	1	Bit 2
cst4	1	0	1	Bit 3
bst1	1	0	0	Bit 4
bst2	0	0	0	Bit 5
bst3	0	0	1	Bit 6
bst4	0	0	1	Bit 7
g0l0				Bit 8
g0l1				Bit 9
g0h0				Bit 10
g0h1				Bit 11
g1t1				Bit 12
g1t3				Bit 13
g2t1				Bit 14
g2t3				Bit 15
g3t1				Bit 16
g3t3				Bit 17
g4t1				Bit 18
g4t3				Bit 19
g5t1				Bit 20
g5t3				Bit 21
redo[0]				Bit 22
redo[1]				Bit 23
loop	0	0	0	Bit 24
exen	0	0	0	Bit 25
amx0	0	0	0	Bit 26
amx1	0	0	0	Bit 27
na	0	0	0	Bit 28
uta	0	0	0	Bit 29
todt	0	0	1	Bit 30
last	0	0	1	Bit 31
	PTS	**PTS+1**	**PTS+2**	

Figure 10-66. Refresh Cycle (CBR) to FPM DRAM



cst1	1	Bit 0
cst2	1	Bit 1
cst3	1	Bit 2
cst4	1	Bit 3
bst1	1	Bit 4
bst2	1	Bit 5
bst3	1	Bit 6
bst4	1	Bit 7
g0l0		Bit 8
g0l1		Bit 9
g0h0		Bit 10
g0h1		Bit 11
g1t1		Bit 12
g1t3		Bit 13
g2t1		Bit 14
g2t3		Bit 15
g3t1		Bit 16
g3t3		Bit 17
g4t1		Bit 18
g4t3		Bit 19
g5t1		Bit 20
g5t3		Bit 21
redo[0]		Bit 22
redo[1]		Bit 23
loop	0	Bit 24
exen	0	Bit 25
amx0	0	Bit 26
amx1	0	Bit 27
na	0	Bit 28
uta	0	Bit 29
todt	1	Bit 30
last	1	Bit 31
	EXS	

Figure 10-67. Exception Cycle

10.5 Initialization/Application Information

These sections provide detailed information on interfacing the LBC to peripheral devices.

10.5.1 Interfacing to Peripherals in Multiplexed Address/Data Mode

The local bus can be used either with separate address- and data buses or with a multiplexed address/data bus. These sections provide guidelines for interfacing peripherals in the multiplexed mode.

10.5.1.1 Multiplexed Address/Data Bus and Unmultiplexed Address Signals

To save signals on the local bus, address and data are multiplexed onto the same 32-bit bus. An external latch is needed to demultiplex and reconstruct the original address. No external intelligence is needed, because LALE provides the correct timing to control a standard logic latch. The LAD signals can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8, 16, and 32 bits. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven unmultiplexed. For 8-bit devices, LA[30:31] should be used and for 16 bit devices, LA30 should be used. 32-bit devices use neither of these signals.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[27:29] are the burst addresses within a natural 32-byte burst. To minimize the amount of address phases needed on the local bus and to optimize the throughput, those signals are driven separately and should be used whenever a device requires the 5 least-significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

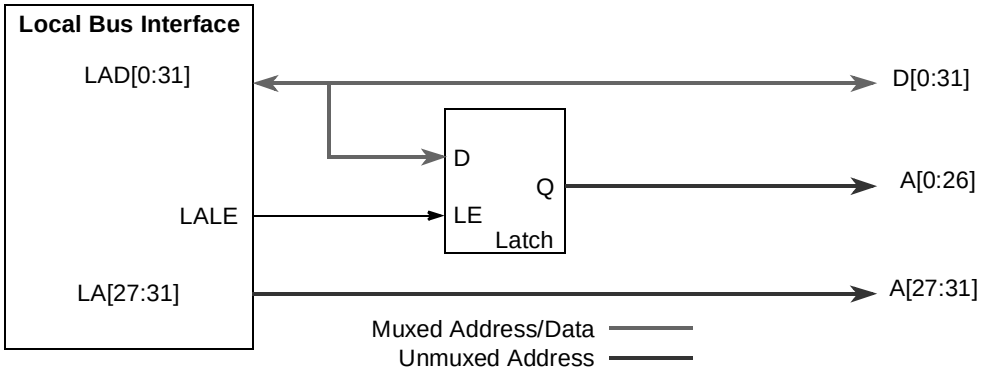


Figure 10-68. Multiplexed Address/Data Bus

10.5.1.2 Peripheral Hierarchy on the Local Bus

To achieve high bus speed interfaces for synchronous SRAMs or SDRAMs, a hierarchy of the memories/peripherals connected to the local bus is suggested, as shown in Figure 10-69.

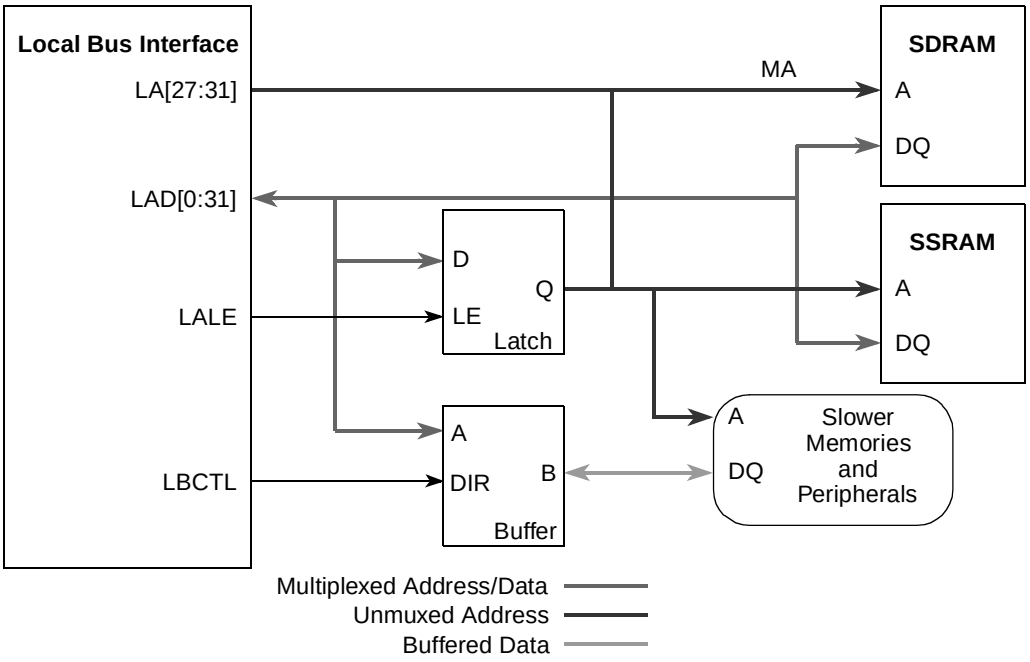


Figure 10-69. Local Bus Peripheral Hierarchy

The multiplexed address/data bus sees the capacitive loading of the data signals of the fast SDRAMs or synchronous SRAMs plus one load for an address latch plus one load for a buffer to the slow memories. The loadings of all other memories and peripherals are hidden behind the buffer and the latch. The system designer needs to investigate the loading scenario and ensure that I/O timings can be met with the loading determined by the connected components.

10.5.1.3 Peripheral Hierarchy on the Local Bus for Very High Bus Speeds

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used. Figure 10-70 shows an example of such a hierarchy. This section is only a guideline; the board designer must simulate the electric characteristics of the chosen scenario to determine the maximum operating frequency.

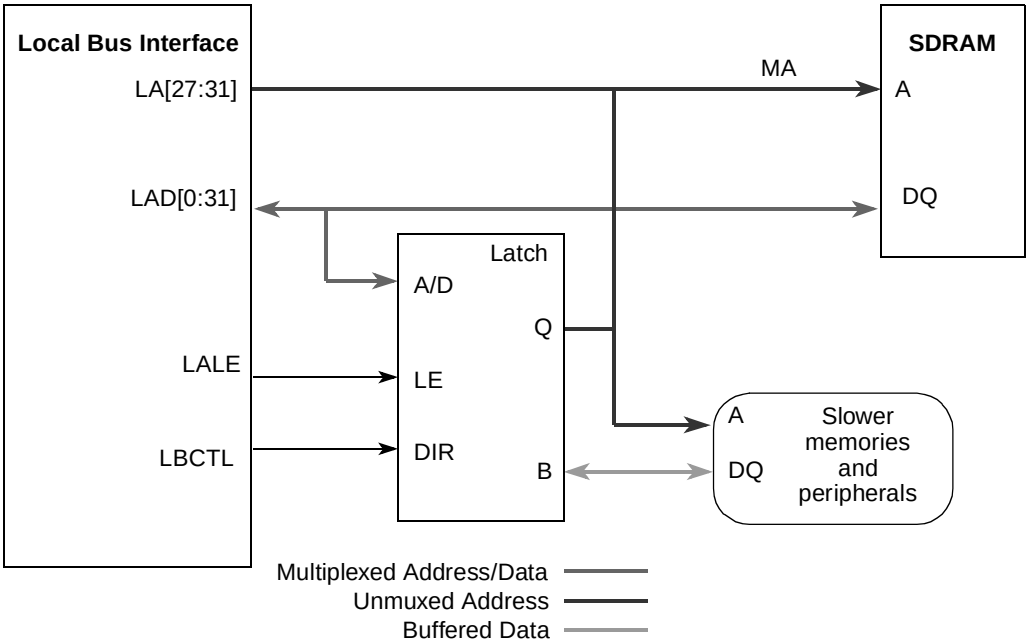


Figure 10-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

10.5.1.4 GPCM Timings

If a system contains a memory hierarchy with high-speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations. The GPCM address timings is shown in Figure 10-71.

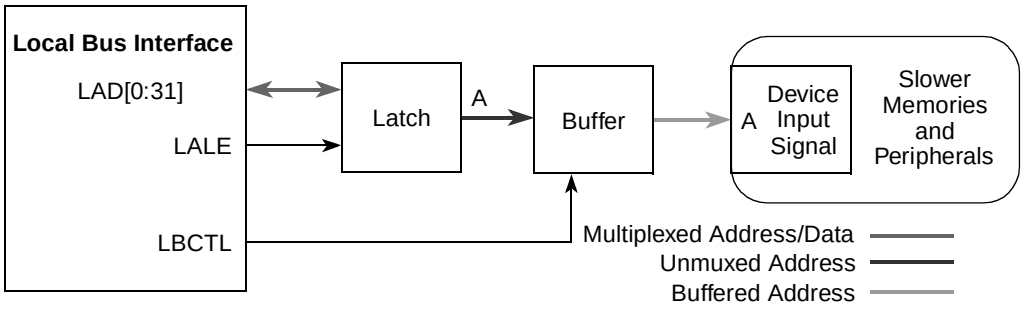


Figure 10-71. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the 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.

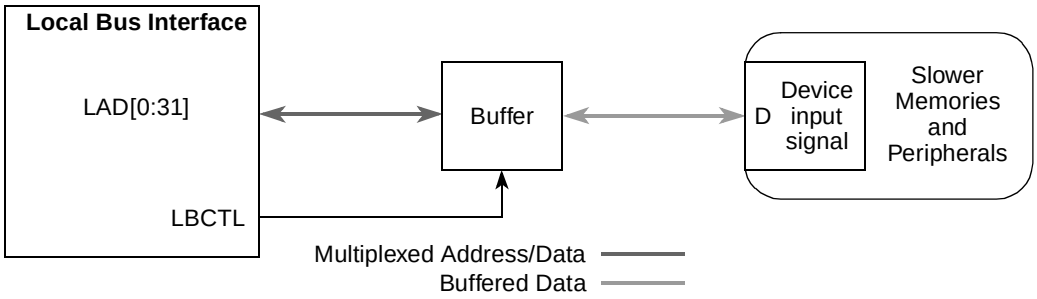


Figure 10-72. GPCM Data Timings

10.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

The bus does not change direction for the following cases, so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

10.5.2.1 Address Phase after Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices, the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The LBC then inserts a bus turnaround cycle to avoid contention. The external device has now already placed its data signals in high impedance and no bus contention will occur.

10.5.2.2 Read Data Phase after Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its t_{dis}(LB). The LBCTL will have its new state after t_{en}(LB) and, because this is an asynchronous input, the transceiver starts to drive those signals after its t_{en}(transceiver) time. The system designer has to ensure, that [t_{en}(LB) + t_{en}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

10.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle has a new address phase in any case, this basically is the same case as an address phase after a previous read.

10.5.2.4 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern. The LBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

10.5.3 Interface to Different Port-Size Devices

The LBC supports 8-, 16-, and 32-bit data port sizes. However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on D[0:31], a 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to transfer the maximum amount of data on all bus cycles. Figure 10-73 shows the device connections on the data bus.

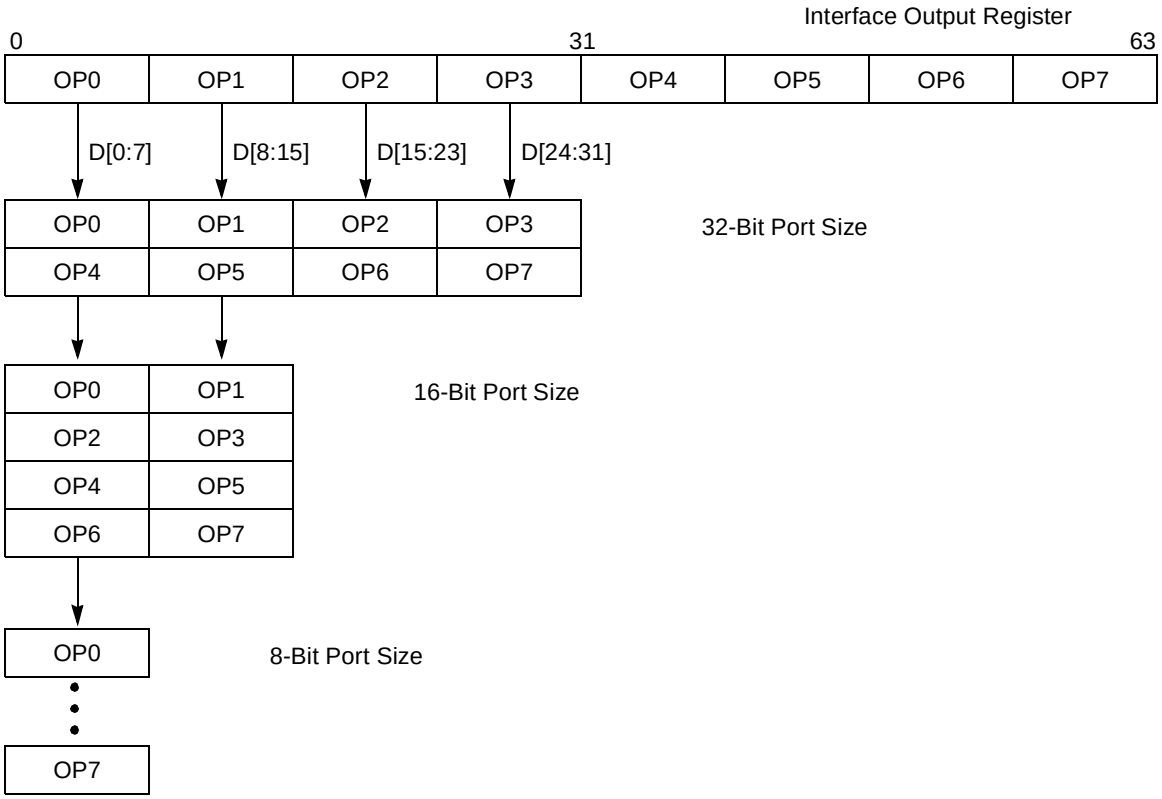


Figure 10-73. Interface to Different Port-Size Devices

Table 10-33 lists the bytes required on the data bus for read cycles.

Table 10-33. Data Bus Requirements For Read Cycle

Transfer Size	Address State [1] A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	0–7	8–15	0–7
Byte	000	OP0[2]	—[3]	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	OP3	OP3
	100	OP4	—	—	—	OP4	—	OP4
	101	—	OP5	—	—	—	OP5	OP5
	110	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	OP7	—	OP7	OP7

Table 10-33. Data Bus Requirements For Read Cycle (continued)

Transfer Size	Address State [1] A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	0–7	8–15	0–7
Half Word	000	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	OP2	OP3	OP2
	100	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	OP6	OP7	OP6	OP7	OP6
Word	000	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	OP4	OP5	OP6	OP7	OP4	OP5	OP4

1 Address state is the calculated address for port size.

2 OP*n*: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.

3 — Denotes a byte not driven during that write cycle.

10.5.4 Interfacing to SDRAM

The following subsections provide application information on interfacing to SDRAM.

10.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects ($\overline{\text{LCS}}n$) can be programmed to support multiple SDRAM devices. Note that no limitation exists on the number of chip selects that can be programmed for SDRAM. This means that $\overline{\text{LCS}}$[1:7] can be programmed to support SDRAM, assuming $\overline{\text{LCS0}}$ is reserved for the GPCM to connect to Flash memory.

If multiple chip selects are configured to support SDRAM on the local bus, each SDRAM device should have the same port size and timing parameters. This means that all option registers (OR*n*) for the SDRAM chip selects should be programmed the same.

NOTE

Although in principle it is possible to mix different port sizes and timing parameters, combinations are limited and this operation is not recommended.

All the chip selects share the same local bus SDRAM mode register (LSDMR) for initialization along with the local bus-assigned SDRAM refresh timer register (LSRT) and the memory refresh timer prescaler register (MPTPR) for refresh.

For refresh, the memory controller supplies auto refresh to SDRAM according to the time interval specified in LSRT and MPTPR as follows:

$$\text{Refresh Period} = \frac{\text{LSRTx (MPTPR[PTP])}}{\text{System Frequency}}$$

This represents the time period required between refreshes. When the refresh timer expires, the memory controller issues a CBR to each chip select. Each CBR is separated by one clock. A refresh timing diagram for multiple chip selects is shown in Figure 10-51 in Section 10.4.3.11.1, "SDRAM Refresh Timing."

During a memory transaction dispatched to the local bus, the memory controller compares the memory address with the address information of each chip select (programmed with BR*n* and OR*n*). If the comparison matches a chip select that is controlled by SDRAM, the memory controller requests service to the local bus SDRAM machine, depending on the information in BR*n*. Although multiple chip selects may be programmed for SDRAM, only one chip select is active at any given time; thus, multiple chip selects can share the same SDRAM machine.

10.5.4.2 Maximum Amount of SDRAM Supported

Table 10-34 summarizes information based on SDRAM data sheets supplied by Micron.

Table 10-34. Micron SDRAM Devices

SDRAM Device	64 Mbit				128 Mbit				256 Mbit				512 Mbit			
I/O Port	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
Bank	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Row	12	12	12	11	12	12	12	12	13	13	13	13	13	13	13	TBD
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	TBD

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires four SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, one chip select provides 128-Mbit/device×4 devices = 64 Mbytes. If 4 chip selects are programmed for SDRAM use, the result is 64 Mbytes×4 = 256 Mbyte. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbyte. Consequently 512-Mbit devices allow for 1 Gbyte.

Although there is no technical difficulty in supporting multiple chip select configurations, in practice, the user may want to maximize the amount of SDRAM assigned to each chip select to minimize cost.

10.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

10.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select Multiplexing

LSDMR[BSMA] is used to multiplex the bank select address. The BSMA field and corresponding multiplexed address are shown below:

```
000 LA12–LA13
001 LA13–LA14
…
111 LA19–LA20
```

Note that LA12 is the latched value of LAD12.

The highest address signals that the bank selects can be multiplexed with are LA[12:13], which limits the signals for the row address to LA[14:31]. For a 32-bit port, the maximum width of the local bus, LA[30:31] are not connected, and the maximum row is LA[14:29]. The local bus SDRAM machine supports 15 rows, which is sufficient for all devices.

10.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address signals to leave room for future upgrades. For example, a user could multiplex the bank select signals to LA[14:15], leaving LA16 to connect to the address signal for a larger memory size.

This allows the system designer to design one board that can be used with a current generation of SDRAM devices and upgraded to the next generation without requiring a new board layout.

10.5.4.3.3 128-Mbyte SDRAM

Figure 10-74 shows the connection to an SDRAM of 128 Mbytes. Note that all circuit diagrams are principal connection diagrams and do not show any means of signal integrity.

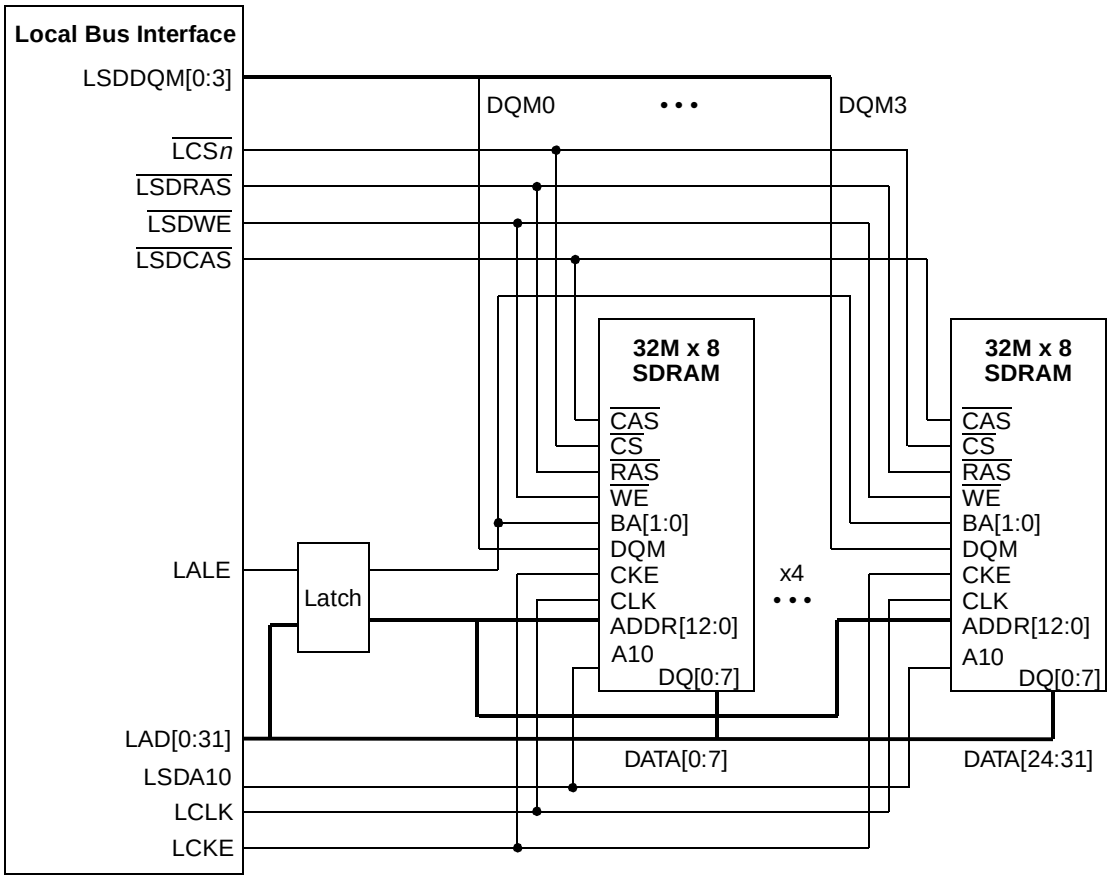


Figure 10-74. 128-Mbyte SDRAM Diagram

Table 10-35 shows details about LAD*n* signal connections for the example in Figure 10-74.

Table 10-35. LAD*n* Signal Connections to 128-Mbyte SDRAM

LAD (Latch Address)	SDRAM Address Signal
LAD29	A0
LAD28	A1
LAD27	A2
LAD26	A3
LAD25	A4
LAD24	A5
LAD23	A6
LAD22	A7

Table 10-35. LAD*n* Signal Connections to 128-Mbyte SDRAM (continued)

LAD (Latch Address)	SDRAM Address Signal
LAD21	A8
LAD20	A9
LAD19 (no connect)	A10 is connected to LSDA10
LAD18	A11
LAD17	A12
LAD16	BA0 (if LSDMR[BSMA] = 011)
LAD15	BA1 (if LSDMR[BSMA] = 011)

Consider the following SDRAM organization:

- The 32-bit port size is organized as 8 x 8 x 32 Mbit.
- Each device has four internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in Table 10-36.

Table 10-36. Logical Address Bus Partitioning

A[0:4]	A[5:17]	A[18:19]	A[20:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters are extracted:

- COLS = 011, 10 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as shown in Table 10-37.

Table 10-37. SDRAM Device Address Port During Address Phase

LA[0:14]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select A[18:19]	Row A[5:17]	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

Table 10-38 shows the address port configuration during a READ/WRITE command.

Table 10-38. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17:18]	LA[19]	LA[20:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

Table 10-39 shows the register configuration for this example. PSRT and MPTPR are not shown but should be programmed according to the device's specific refresh requirements.

Table 10-39. Register Settings for 128-Mbytes SDRAMs

Register	Field	Value
BR*n*	BA	Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR*n*	AM	11_1111_1000_0000_0000_0
	COLS	011
	ROWS	100
LSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

10.5.4.3.4 256-Mbyte SDRAM

This example uses the same SDRAM as the previous example, but doubles the number of devices connected and therefore uses two chip selects.

10.5.4.3.5 512-Mbyte SDRAM

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is 8 x 4 x 64 Mbit x 2 chip select lines.
- Each device has 4 internal banks, 13 row address lines, and 11 column address lines.

The logical address is partitioned as shown in Table 10-40.

Table 10-40. Logical Address Partitioning

A[0:3]	A[4:16]	A[17:18]	A[19:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- COLS = 100, 11 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as in Table 10-41.

Table 10-41. SDRAM Device Address Port During Address Phase

LA[0:13]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select (A[17:18])	Row (A[4:16])	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

Table 10-42 shows the address port settings during a READ/WRITE command.

Table 10-42. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17]	LA[18]	LA[19:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

Table 10-43 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 10-43. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA	Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR*n*	AM	11_1110_0000_0000_0000_0
	BPD	01
	COLS	100
	ROWS	100
PSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

10.5.4.4 Parity Support for SDRAM

Unlike older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16 or x32. No mainstream devices include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Because the local bus allows for SDRAM accesses with less than the full port size, read-modify-write cycles are supported for SDRAM write cycles.

Figure 10-75 shows a connection diagram.

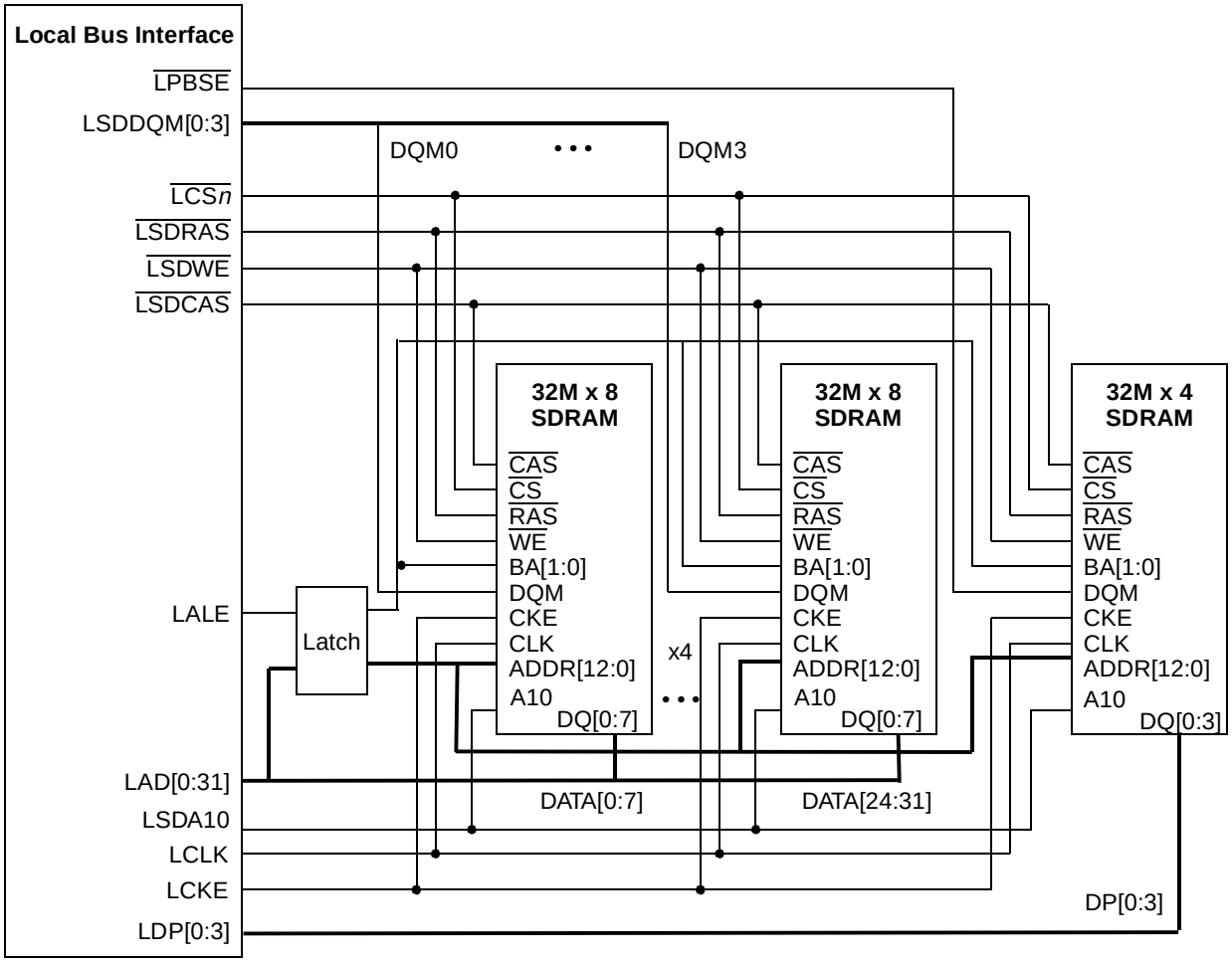


Figure 10-75. Parity Support for SDRAM

10.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, memory access patterns are often random and SDRAM is not optimized for that case. ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 10-76 shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs are used mostly by performance-critical applications, a 32-bit maximum local bus width is assumed.

ZBT SRAMs allow different configurations. For the local bus, the burst order should be set to linear burst order by tying the mode signal to GND; $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates eight-beat transactions (for 32-bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the LBC generates the new A27 for the second burst.

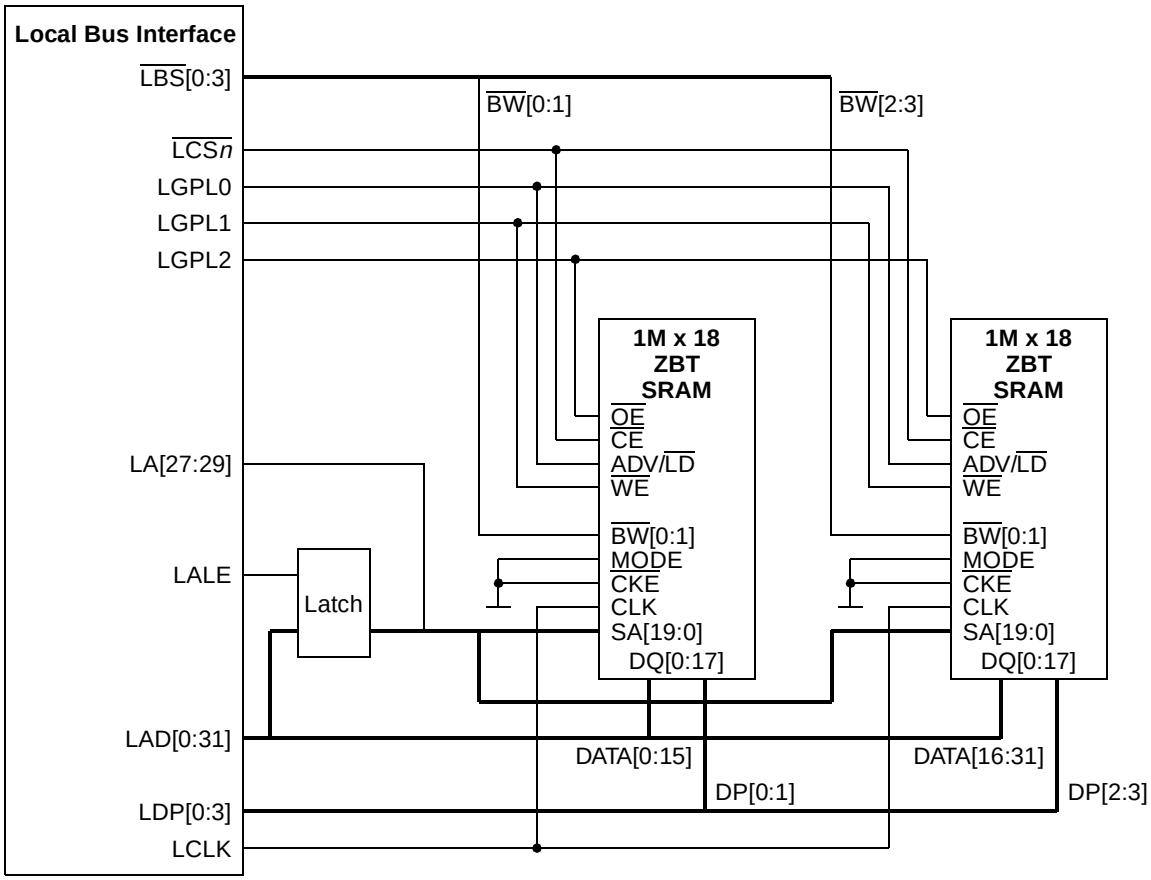


Figure 10-76. Interface to ZBT SRAM

Because linear burst on the SRAM is used, the device itself bursts with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7]. Therefore, two consecutive

linear bursts of the ZBT SRAM with A27 = 0 for the first burst and A27 = 1 for the second burst give the desired burst pattern.

The UPM also supports single beat accesses. Because the ZBT SRAM does not support this and always responds with a burst, the UPM pattern must take care that data for the critical beat is provided (for write) or sampled (for read), and that the rest of the burst is ignored (by negating $\overline{WE}$). The UPM controller must wait for the end of the SRAM burst to avoid bus contention with further bus activities.

Note that SRAMs are available with natural parity. In the example, a ×18 SRAM, which holds two data bytes and two parity bits, is used. While for the support of parity on SDRAM banks, the local bus must use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

10.5.5.1 Interfacing to MSC8122 DSI

The MSC8122 direct-slave interface (DSI) gives an external host direct access to the MSC8122. It provides the following slave interfaces to an external host:

- Asynchronous SRAM-like interface giving the host single accesses (with no external clock).
- Synchronous SSRAM-like interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8122 internal bus clock.

The DSI supports 32- or 64-bit data bus modes. For connection to the local bus, the DSI must be configured in 32-bit mode. As part of the DSP reset.

The DSI supports two addressing modes, which are determined during the MSC8122 boot sequence and are used in both 32- and 64-bit data modes. Refer to details in the MSC8122 documentation.

- Full address bus mode with HA[11–29]
- Sliding window mode with HA[14–29]

10.5.5.2 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single- and double-strobe of operation. The dual strobe configuration in Figure 10-77 shows the interface to the MSC8122 DSI for asynchronous mode.

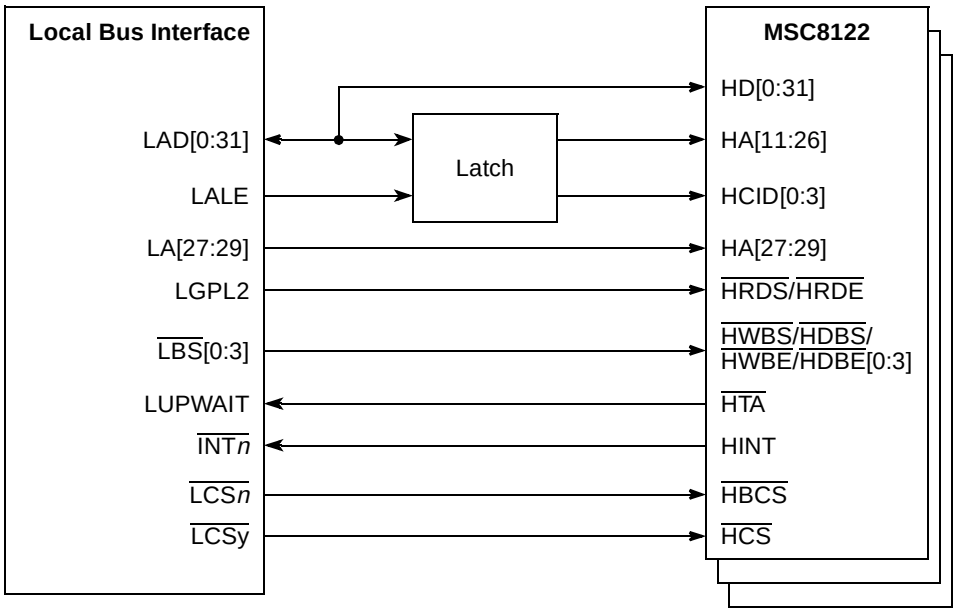


Figure 10-77. Interface to MSC8122 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used if only relatively small amounts of data are transferred between the communications controller and the MSC8122. For maximum timing flexibility, the UPM machine of the LBC should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. The UPM allows sufficient control to satisfy any requirements here.

Figure 10-78 shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0:3]) on the first falling edge of the host write byte strobe signals ($\overline{\text{HWBS}}$) on which the host chip select signal ($\overline{\text{HCS}}$) is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. By asserting $\overline{\text{HTA}}$, the DSI signals whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating $\overline{\text{HWBS}}$. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. M*n*MR[UWPL] must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of $\overline{\text{HWBS}}$. In addition, the assertion of $\overline{\text{HWBS}}$[0:3] are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[HTAAD] should be set 1 and DCR[HTADT] should be defined to a non zero value. This mode is to be used in implementations with a pull-up resistor on $\overline{\text{HTA}}$. The host can start its next access (back-to-back accesses) without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, to prevent contention on $\overline{\text{HTA}}$, the host must wait until the previous DSI stops driving $\overline{\text{HTA}}$ before it accesses the next device. If the next access is to the same MSC8122, the host must not start consecutive accesses before $\overline{\text{HTA}}$ is asserted by the previous access. The

easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

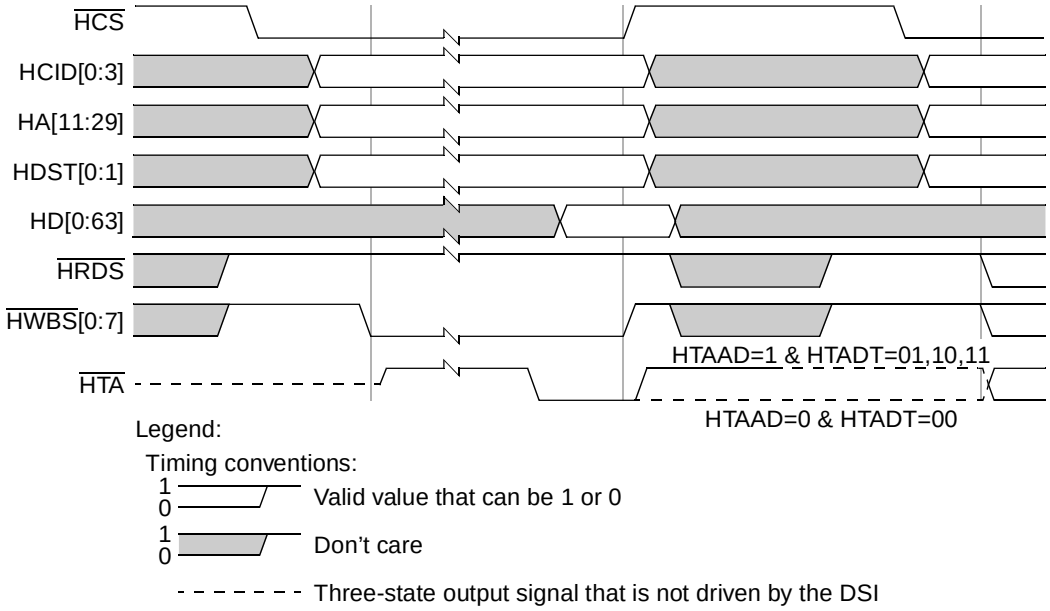


Figure 10-78. Asynchronous Write to MSC8122 DSI

Figure 10-79 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ($\overline{\text{HRDS}}$) on which the $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. When the DCR[RPE] bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. By asserting $\overline{\text{HTA}}$, the DSI signals that the data is valid and that the host can sample the host data bus (HD) and terminate the access by negating $\overline{\text{HRDS}}$. If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of $\overline{\text{HTA}}$ is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. M*n*MR[UWPL] has to be cleared to interpret the correct polarity of the $\overline{\text{HTA}}$ signal.

Because the UPM is used in the mode, the DCR[HTAAD] should be set 1 and the drive time control field, DCR[HTADT], should be defined to a non zero value. This mode is specially designed to be used for implementations with a pull-up resistor on $\overline{\text{HTA}}$.

The host can start its next access (back-to-back accesses) without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, then to prevent contention on $\overline{\text{HTA}}$, the host must wait until the previous DSI stops driving $\overline{\text{HTA}}$ before it accesses the next device. If the next access is to the same MSC8122, the host must not start consecutive access before $\overline{\text{HTA}}$ is actively driven to 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

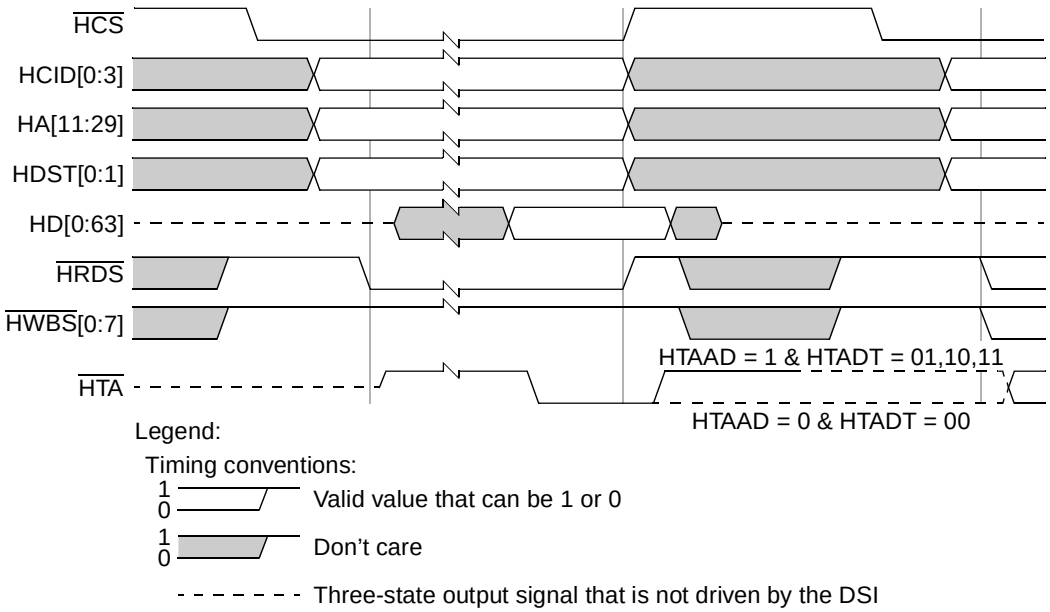


Figure 10-79. Asynchronous Read from MSC8122 DSI

10.5.5.3 DSI in Synchronous Mode

The synchronous SSRAM-like mode of the DSI is inherently faster than the asynchronous mode and should be used if larger amounts of data are transferred between the communications controller and the MSC8122. This will optimize the bus utilization, especially if several MSC8122s are connected to one local bus. The UPM machine of the LBC must be used to implement this interface.

Figure 10-80 shows the interface for synchronous mode. Because the DSI asserts and negates $\overline{\text{HTA}}$ in synchronous mode even within a burst transfer on a clock-by-clock basis, and because the DSI expects the host to react within one clock cycle, some tricks can be implemented to support the synchronous mode.

$\overline{\text{HTA}}$ drives LUPWAIT of the UPM. M*n*MR[UWPL] must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to $\overline{\text{HTA}}$ changes correctly within one local bus clock. Refer to Section 10.4.4.4.10, "Wait Mechanism (WAEN)."

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3 or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3, or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1x, 2x, 3x, or 4x, which should be sufficient for any likely divider ratio.

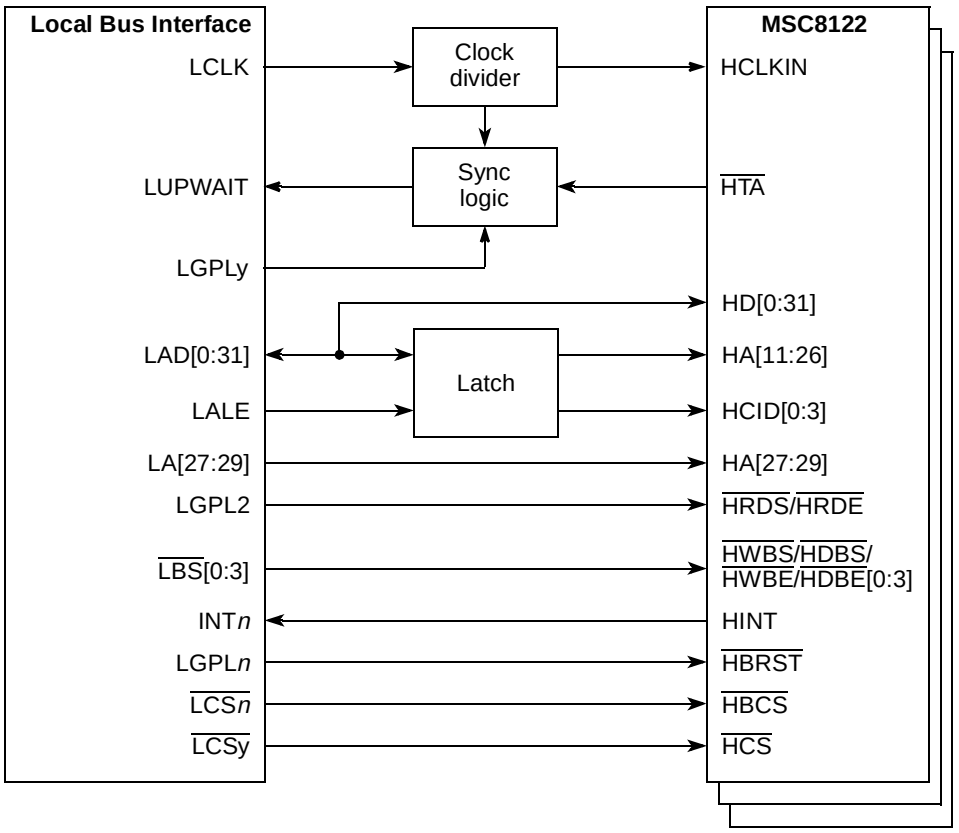


Figure 10-80. Interface to MSC8122 DSI in Synchronous Mode

This solution allows the local bus to react within multiple local bus clocks to the $\overline{\text{HTA}}$ signal and be still within one DSI clock. Typically, LUPWAIT is synchronized internal, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is 3x the DSI clock, data can be sampled in the third local bus sub-clock, which is the last third of the DSI clock. If the local bus clock ratio is only 2x the DSI clock, there is a special mode, where the LUPWAIT is not synchronized. Refer to Section 10.4.4.5, "Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge." In this mode, data is sampled in the 2nd subclock, which is the second half of the DSI clock. AC timing of LUPWAIT must be met in this mode; otherwise behavior is indeterminate.

The remaining issue is the synchronization of the UPM cycles to the beginning of the DSI clock cycle. Because the UPM executes *n* cycles for every DSI cycle, UPM transitions must be synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal controls a multiplexer and activates external synchronization logic, which uses the DSI clock to stall the UPM by asserting LUPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexer connects LUPWAIT to $\overline{\text{HTA}}$ instead, for the rest of the bus cycle. Note that the GPL signals should be used in the inverted state of their inactive state (GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive) to start the synchronization process.

Figure 10-81 shows such a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary from one to *n* (clock ratio) local bus clocks.

The second column (compensation cycle) is intended to compensate for the reaction time of LUPWAIT to get in lock step with the DSI clock. For example, if the clock divider ratio is 1:3 and the LUPWAIT reaction time is two local bus clocks, because LUPWAIT is synchronized, then one local bus clock should be inserted.

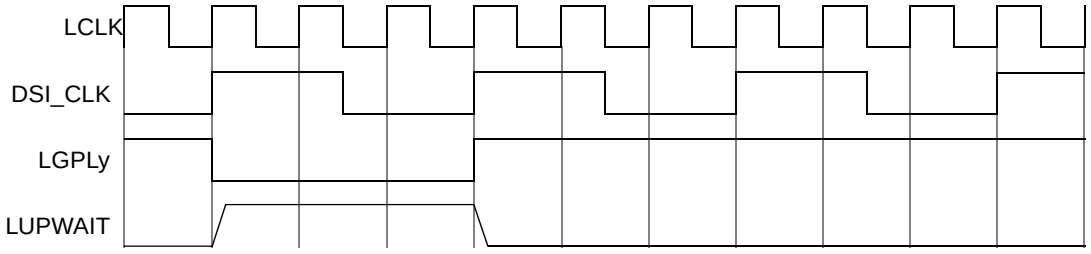


Figure 10-81. UPM Synchronization Cycle

Table 10-44. UPM Synchronization Cycles

	Synchronization Cycle	Compensation Cycle	DSI Cycle 1	Bits
cst1–cst4				0–3
bst1–bst4				4–7
g0xx				8–11
g1tx				12–13
g2tx				14–15
g3tx				16–17
g4t1				18
g4t3	1	0		19
g5tx				20–21
redo[0]	0	0	1	22
redo[1]	0	0	0	23
loop	0	0		24
exen	0			25
amx0	0	0		26
amx1	0	0		27
na	0			28
uta	0			29
todt	0			30
last	0			31

This section describes synchronous single write and read, and synchronous burst write and read operations.

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. The dual strobe configuration is shown as an example.

10.5.5.3.1 Synchronous Single Write

Figure 10-82 shows a synchronous single write access.

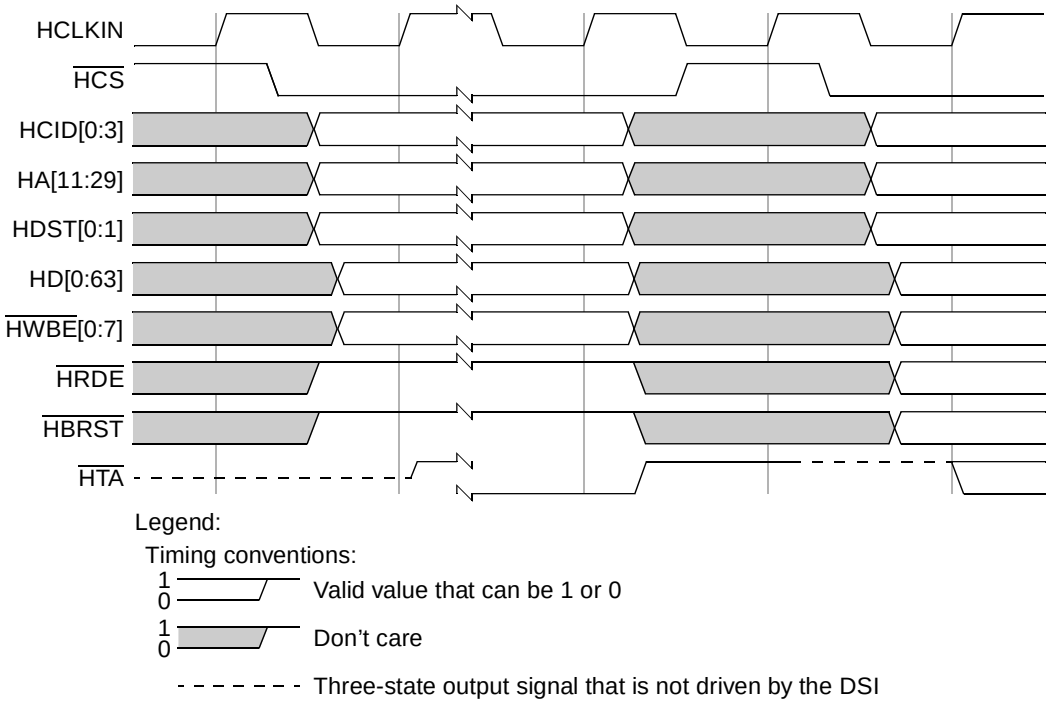


Figure 10-82. Synchronous Single Write to MSC8122 DSI

The DSI samples HA, HDST, HCID, HD, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. At least one HWBE signal is asserted, and HRDE and HBRST are negated. Assertion of HTA indicates that the DSI is ready to complete the current access and the host must terminate this access. Because HTA is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. Typically, HTA is asserted immediately. If the write buffer is full, HTA assertion is delayed. HTA is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately on the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8122, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

10.5.5.3.2 Synchronous Single Read

Figure 10-83 shows a synchronous single read access.

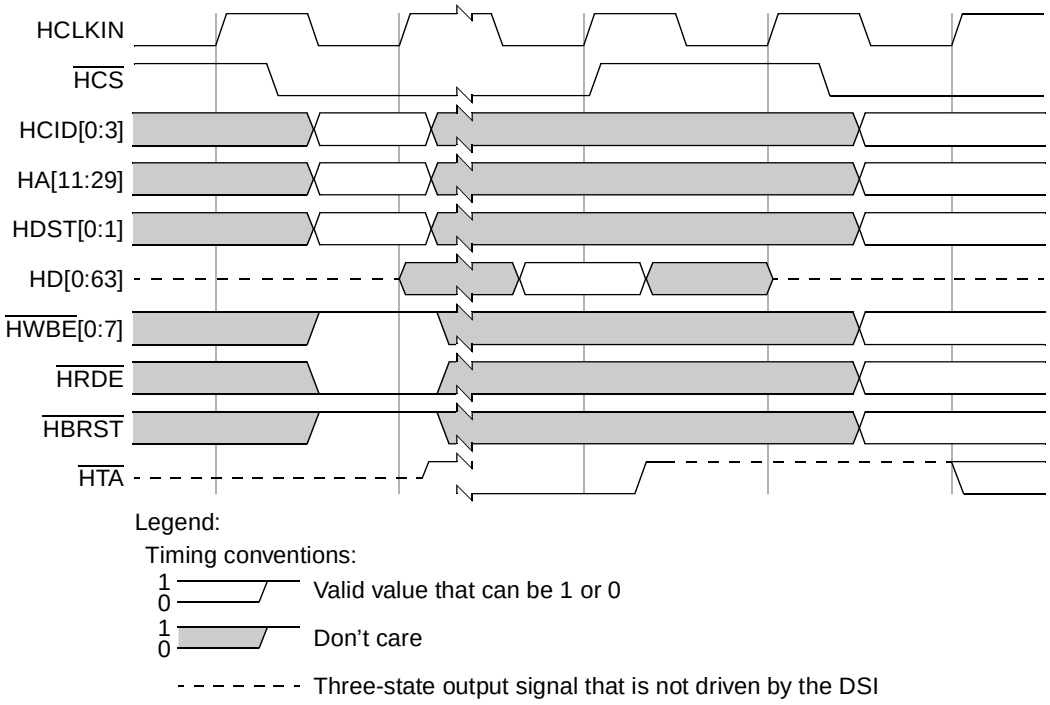


Figure 10-83. Synchronous Single Read from MSC8122 DSI

The DSI samples HA, HDST, HCID, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. HRDE is asserted, and HWBE and HBRST are negated. If DCR[RPE] is set (see MSC8122 documentation), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of HTA indicates that data is valid and the host must sample the HD and terminate the access. Because HTA is connected to the UPM LUPWAIT signal, all local bus signals are frozen until HTA goes to 0; then the UPM continues in its pattern. HTA is asserted earlier when the data for this access is already prefetched to the read buffer. It asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8122, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

10.5.5.3.3 Synchronous Burst Write

Figure 10-84 shows a synchronous burst write access.

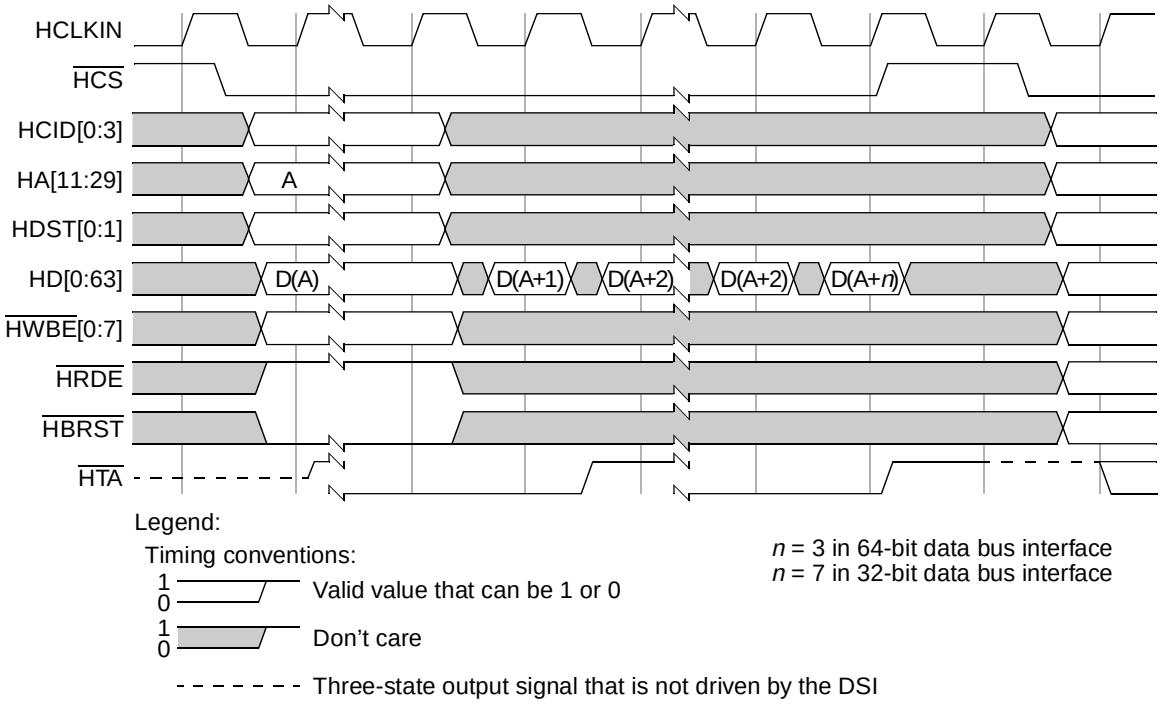


Figure 10-84. Synchronous Burst Write to MSC8122 DSI

The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. $\overline{\text{HWBE}}$ are asserted, $\overline{\text{HBRST}}$ is asserted, and $\overline{\text{HRDE}}$ is negated. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically $\overline{\text{HTA}}$ is asserted immediately for each beat of the access. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. After the last beat of the access, $\overline{\text{HTA}}$ is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

10.5.5.3.4 Synchronous Burst Read

Figure 10-85 shows a synchronous burst read access. The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.

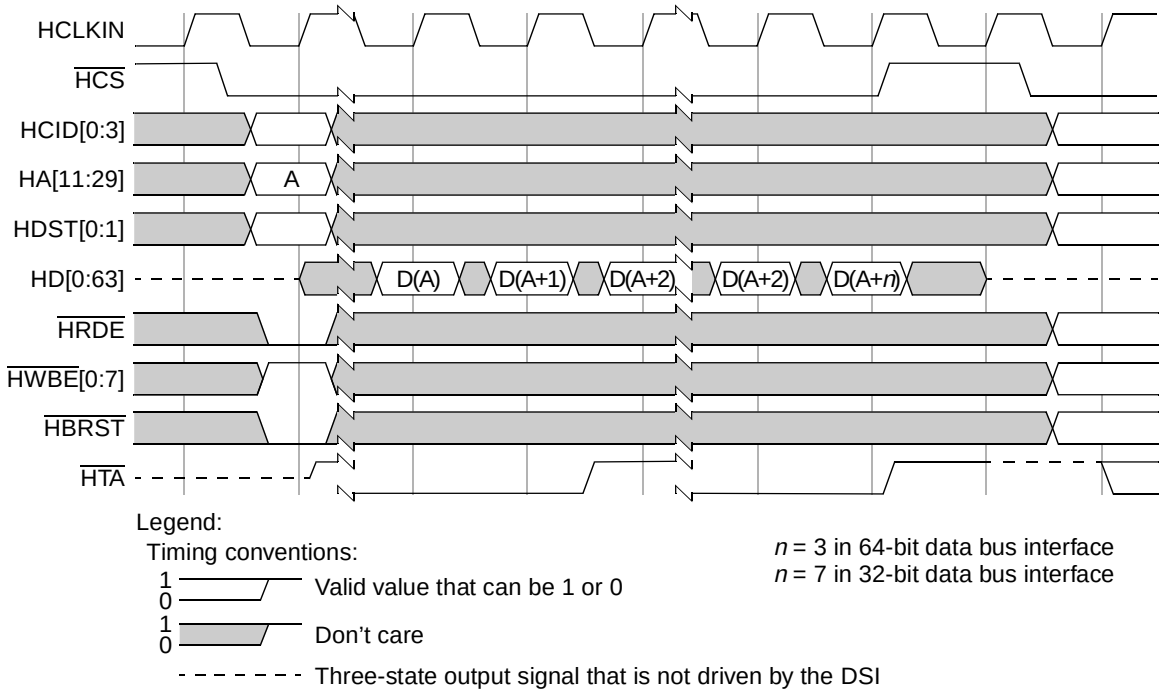


Figure 10-85. Synchronous Burst Read from MSC8122 DSI

$\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When DCR[RPE] (see the MSC8122 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8122 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8122, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

10.5.5.4 Broadcast Accesses

By using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8122 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode HCID[0:3].

Note that broadcasting is allowed only for write accesses.

The DSI sets the overflow bit, DER[OVF], if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it. To avoid overflow when accessing DSI registers during broadcast accesses, wait at least 10 host clock cycles in synchronous mode or 8 internal clock cycles in asynchronous mode between each DSI register access.

To avoid data corruption, if DER[OVF] is set, no broadcast access is written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, DER[OVF] must first be read and reset if it is set.

NOTE

In asynchronous mode, write data from a previous access (even a normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers, depending on the type of previous access. This can be achieved by performing a read access before the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode, $\overline{\text{HWBS}}$[0:3]/$\overline{\text{HDBS}}$[0:3] assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8122 Technical Data* sheet.
- In synchronous mode single access, the host must wait 1 cycle before terminating the access. Access signals must be in the same valid state during two positive edges of the host clock cycles. Access duration is two clock cycles (the DSI may translate accesses lasting longer than two clock cycles as two or more back-to-back accesses).
- In synchronous mode burst accesses, broadcast accesses are not allowed.

Chapter 11
Sequencer

11.1 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).

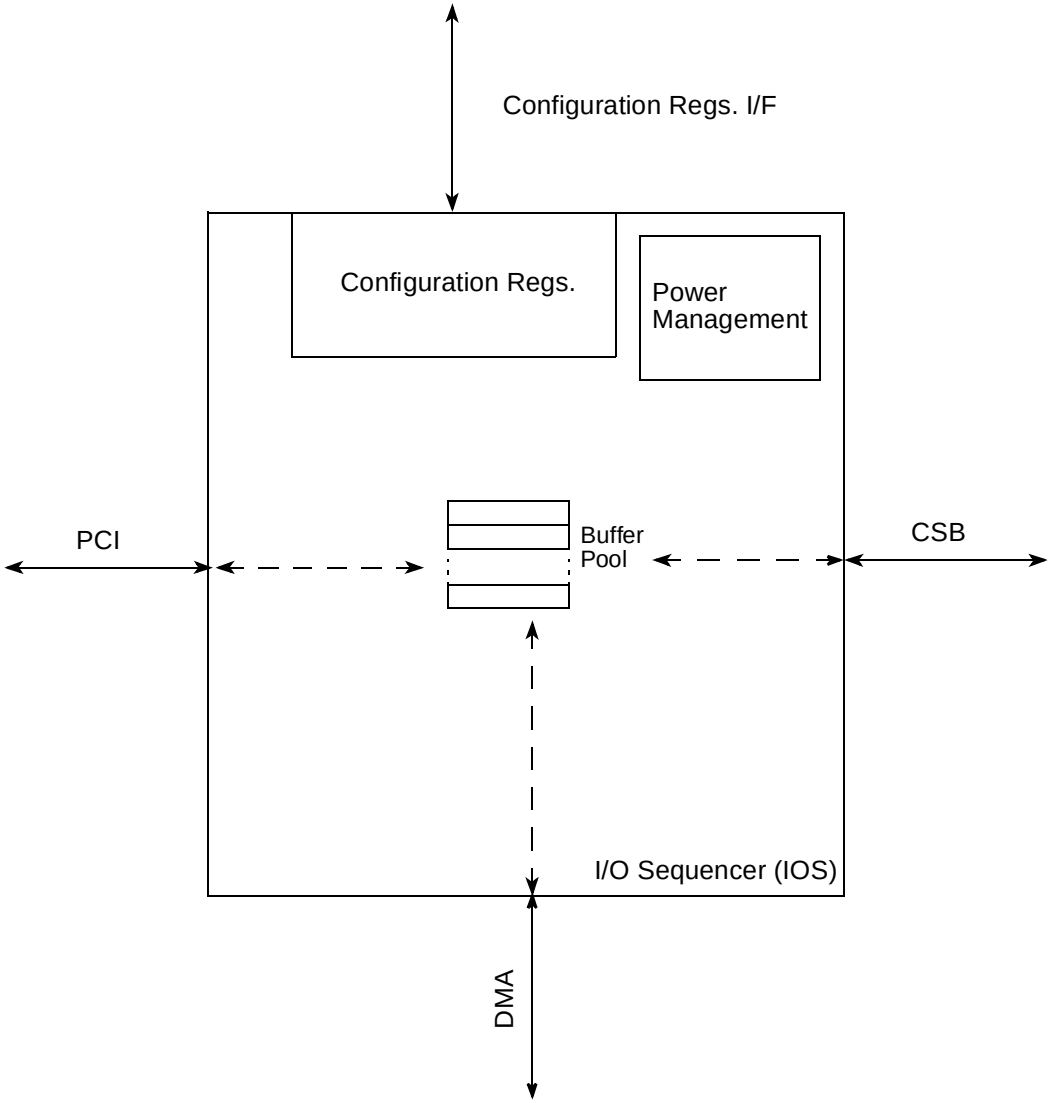


Figure 11-1. I/O Sequencer Block Diagram

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.

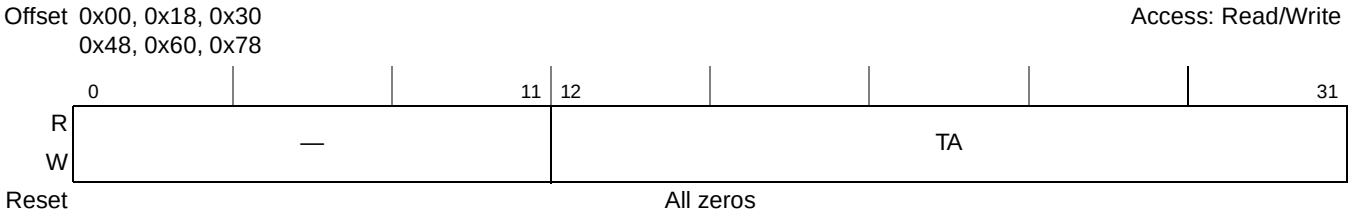


Figure 11-2. PCI Outbound Translation Address Registers (POTAR*n*)

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.

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.

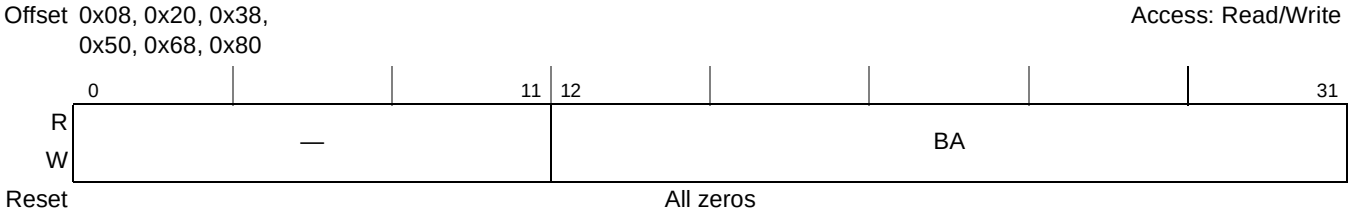


Figure 11-3. PCI Outbound Base Address Registers (POBAR*n*)

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.

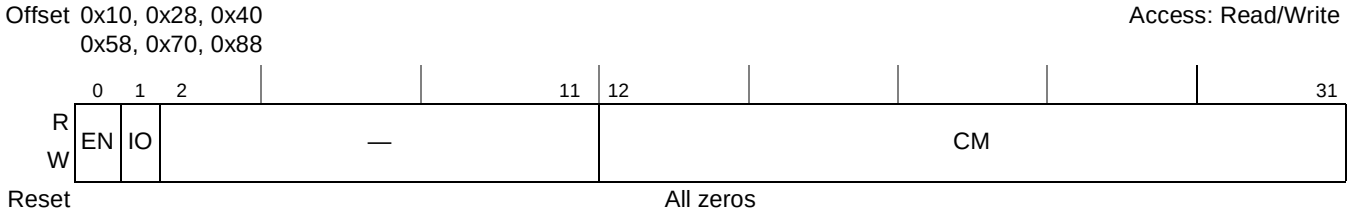


Figure 11-4. PCI Outbound Comparison Mask Registers (POCMR*n*)

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.

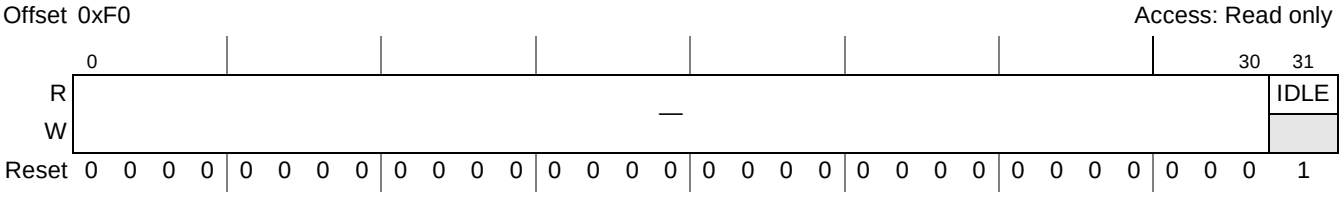


Figure 11-5. Power Management Control Register (PMCR)

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 <<BLOCK NAME>> 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.

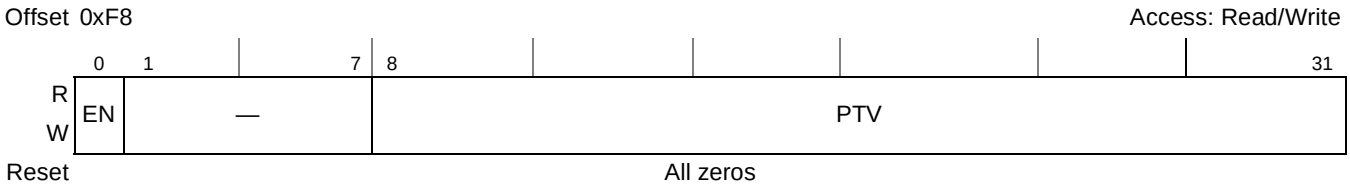


Figure 11-6. Discard Timer Control Register (DTCR)

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.

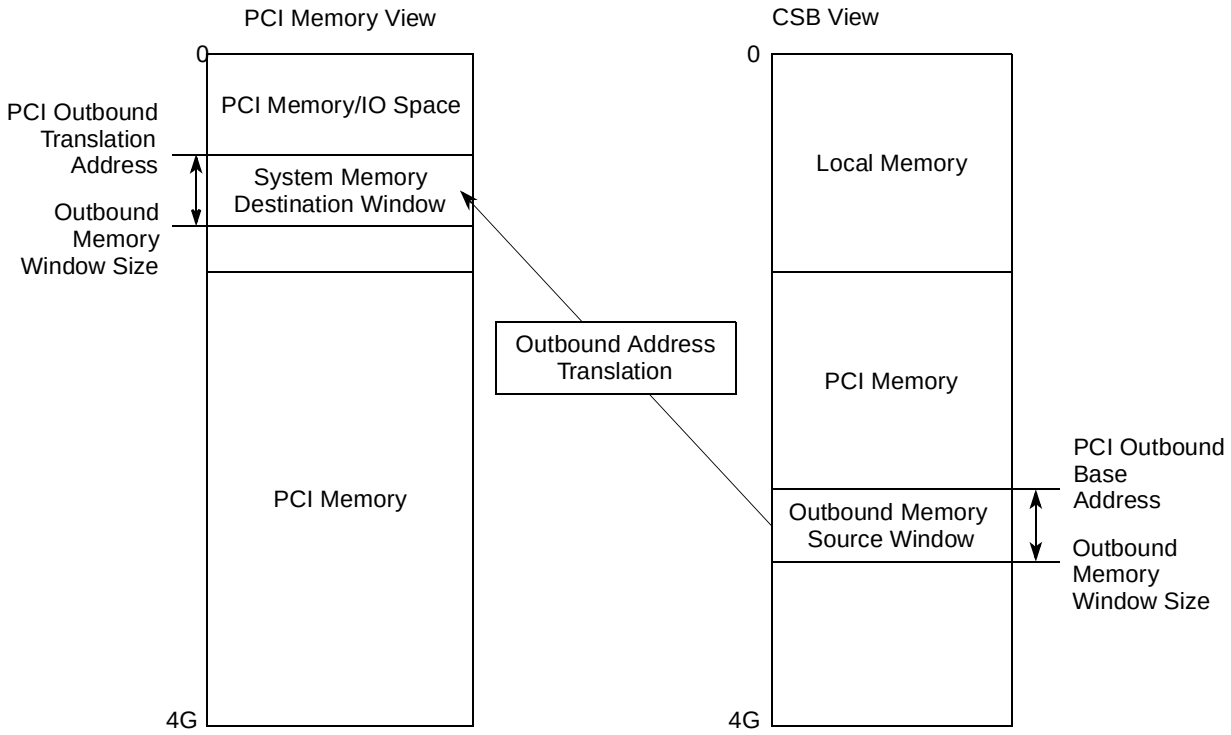


Figure 11-7. Outbound PCI Memory Address Translation

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.

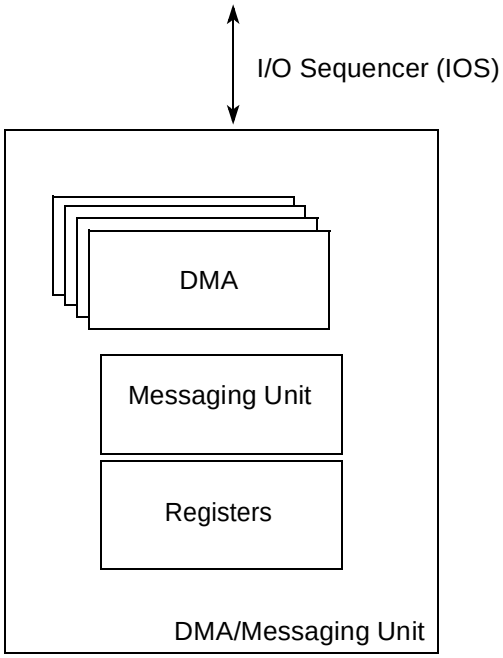


Figure 12-1. DMA/Messaging Unit Block Diagram

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
  - — Optional external control signals (REQ/ACK/DONE) per channel

12.2 External Signal Description

This section describes the DMA signals.

12.2.1 Detailed Signal Descriptions

Table 12-1 contains the detailed descriptions of the DMA interface signals.

Table 12-1. DMA Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DREQ}}$[0:3]	I	DMA request signals, one per channel. The DMA request signal indicates the start or continuation of a DMA transfer. The falling edge of $\overline{\text{DREQ}}n$ causes DMAMR*n*[CS] to be set, thereby activating the DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DREQ}}n$ starts or resumes a DMA transfer if DMAMR*n*[EMSEN] is 1. Negated—Negation of $\overline{\text{DREQ}}n$ has no effect.
		Timing	Assertion—Can be asserted asynchronously. Negation—Should remain asserted until $\overline{\text{DACK}}n$ is asserted or the requested transaction to the peripheral occurs.
$\overline{\text{DACK}}$[0:3]	O	DMA acknowledge signals, one per channel. The DMA acknowledge signal reflects the value of DMAMR*n*[CS].	
		State Meaning	Asserted—A DMA transfer is active. Negated—The DMA transfer is halted or complete.
		Timing	Assertion—Asserted asynchronously when a DMA transfer is started or resumed in the internal control logic. Negation—Negated asynchronously when a DMA transfer is halted or completed in the internal control logic. Note that there may still be outstanding write transactions in the bus pipeline after the negation of $\overline{\text{DACK}}n$.

Table 12-1. DMA Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{DDONE}}$[0:3]	O	DMA done signals, one per channel. The DMA done signal indicates that the DMA transfer has completed.	
		State Meaning	Asserted—A DMA transfer is complete. Negated—A DMA transfer is active or halted.
		Timing	Assertion—Asserted asynchronously when a DMA transfer is completed in the internal control logic. Note that there may still be outstanding write transactions in the bus pipeline after the assertion of $\overline{\text{DDONE}}n$. Negation—Negated asynchronously when a DMA transfer begins in the internal control logic.

12.3 Memory Map/Register Definition

Table 12-2 lists the address and access of the memory map module.

Table 12-2. Module Memory Map

Offset	Register	Access	Reset	Section/Page
0x00_8030	OMISR—Outbound message interrupt status register	special	0x0000_0000	12.4.1/12-4
0x00_8034	OMIMR—Outbound message interrupt mask register	R/W	0x0000_0000	12.4.2/12-6
0x00_8050	IMR0—Inbound message register 0	R/W	0x0000_0000	12.4.3/12-7
0x00_8054	IMR1—Inbound message register 1	R/W	0x0000_0000	12.4.3/12-7
0x00_8058	OMR0—Outbound message register 0	R/W	0x0000_0000	12.4.4/12-7
0x00_805C	OMR1—Outbound message register 1	R/W	0x0000_0000	12.4.4/12-7
0x00_8060	ODR—Outbound doorbell register	R/W	0x0000_0000	12.4.5/12-8
0x00_8068	IDR—Inbound doorbell register	R/W	0x0000_0000	12.4.5/12-8
0x00_8080	IMISR—Inbound message interrupt status register	Mixed	0x0000_0000	12.4.6/12-9
0x00_8084	IMIMR—Inbound message interrupt mask register	R/W	0x0000_0000	12.4.7/12-11
0x00_8100	DMAMR0—DMA 0 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x00_8104	DMASR0—DMA 0 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x00_8108	DMACDAR0—DMA 0 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x00_8110	DMASAR0—DMA 0 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x00_8118	DMADAR0—DMA 0 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x00_8120	DMABCR0—DMA 0 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x00_8124	DMANDAR0—DMA 0 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x00_8180	DMAMR1—DMA 1 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x00_8184	DMASR1—DMA 1 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x00_8188	DMACDAR1—DMA 1 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15

Table 12-2. Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x00_8190	DMASAR1—DMA 1 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x00_8198	DMADAR1—DMA 1 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x00_81A0	DMABCR1—DMA 1 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x00_81A4	DMANDAR1—DMA 1 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x00_8200	DMAMR2—DMA 2 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x00_8204	DMASR2—DMA 2 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x00_8208	DMACDAR2—DMA 2 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x00_8210	DMASAR2—DMA 2 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x00_8218	DMADAR2—DMA 2 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x00_8220	DMABCR2—DMA 2 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x00_8224	DMANDAR2—DMA 2 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x00_8280	DMAMR3—DMA 3 mode register	R/W	0x0000_0000	12.4.8.1/12-11
0x00_8284	DMASR3—DMA 3 status register	R/W	0x0000_0000	12.4.8.2/12-14
0x00_8288	DMACDAR3—DMA 3 current descriptor address register	R/W	0x0000_0000	12.4.8.3/12-15
0x00_8290	DMASAR3—DMA 3 source address register	R/W	0x0000_0000	12.4.8.4/12-16
0x00_8298	DMADAR3—DMA 3 destination address register	R/W	0x0000_0000	12.4.8.5/12-16
0x00_82A0	DMABCR3—DMA 3 byte count register	R/W	0x0000_0000	12.4.8.6/12-17
0x00_82A4	DMANDAR3—DMA 3 next descriptor address register	R/W	0x0000_0000	12.4.8.7/12-17
0x00_82A8	DMAGSR—DMA general status register	R	0x0000_0000	12.4.8.8/12-18
0x00_82B0–0x00_82FF	Reserved	—	—	—

12.4 Register Descriptions

The following sections describe the DMA/messaging unit configuration, control, and status registers.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

12.4.1 Outbound Message Interrupt Status Register (OMISR)

OMISR contains the interrupt status of the doorbell and outbound message registers. A PCI device acknowledges the outbound message interrupt by writing a 1 to the appropriate status bit: OMISR[OM1I] or OMISR[OM0I]. Setting one of these bits clears both the interrupt and the corresponding status bit. The

local processor provokes an outbound message interrupt by writing to either of the two outbound message registers: OMR0 or OMR1. OMISR can be accessed from the CSB or the PCI bus, but it is normally accessed only from the PCI bus. Figure 12-2 shows the OMISR fields.

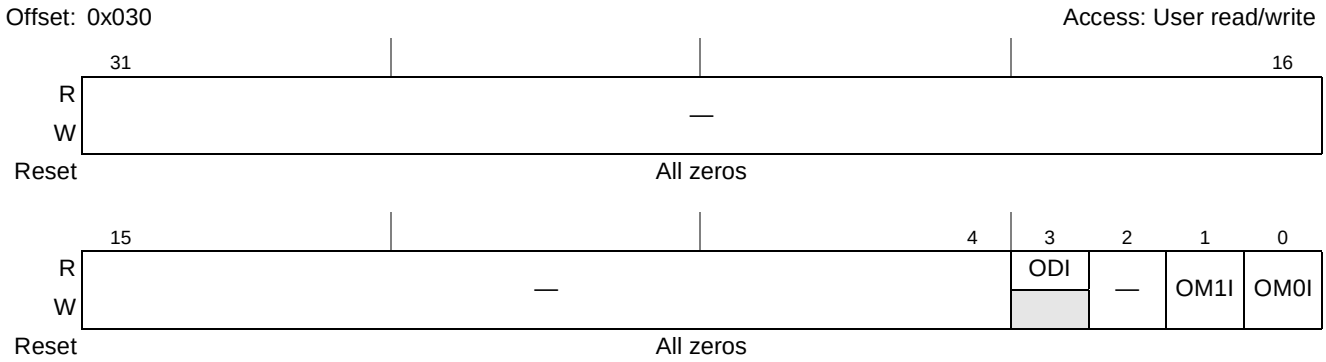


Figure 12-2. Outbound Message Interrupt Status Register (OMISR)

Table 12-3 describes the OMISR register.

Table 12-3. 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.

12.4.2 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 cleared only from the PCI bus. Figure 12-3 shows the OMIMR.

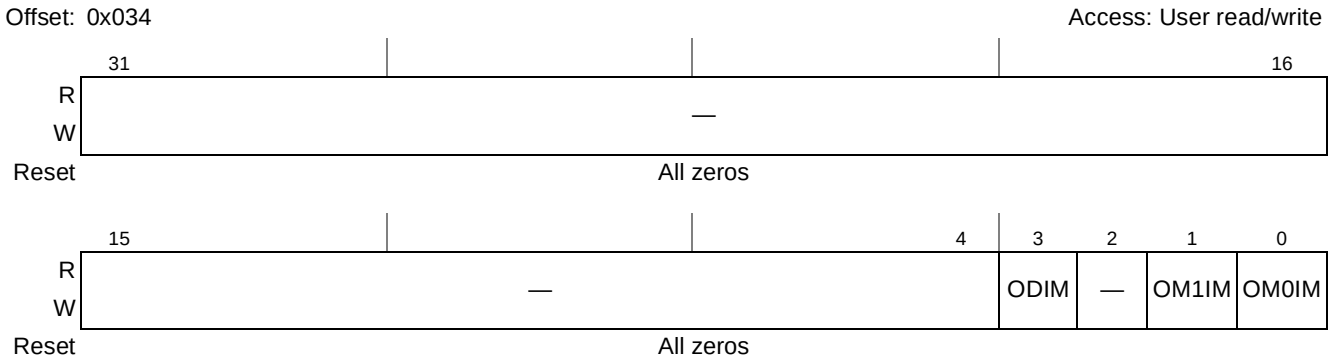


Figure 12-3. Outbound Message Interrupt Mask Register (OMIMR)

Table 12-4 describes the OMIMR register.

Table 12-4. 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

12.4.3 Inbound Message Registers

The inbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the PCI bus. Figure 12-4 shows the IMR0 and IMR1 fields.

Figure 12-4. Inbound Message Registers (IMR0, IMR1)

Table 12-5 describes the IMR*n* register.

Table 12-5. IMR0 and IMR1 Field Descriptions

Bits	Name	Description
31–0	IMSG*n*	Inbound message *n*. Contains generic data to be passed between the local processor and external hosts.

12.4.4 Outbound Message Registers (OMR0–OMR1)

The outbound message registers can be read from the PCI bus and the CSB in both host and agent modes. They can be written only from the CSB.

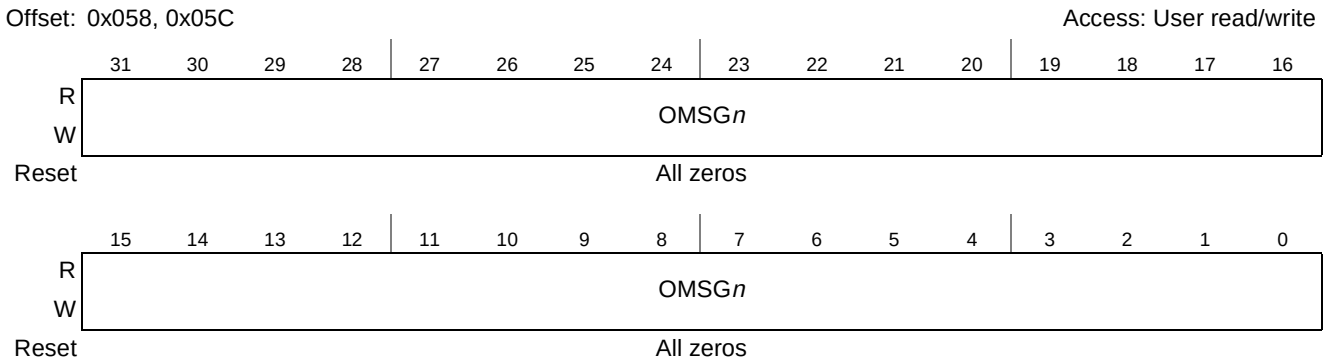


Figure 12-5. Outbound Message Registers (OMR0–OMR1)

Table 12-6 describes the OMR*n* registers.

Table 12-6. OMR0 and OMR1 Field Descriptions

Bits	Name	Description
31–0	OMSG*n*	Outbound message *n*. Contains generic data to be passed between the local processor and external hosts.

12.4.5 Doorbell Registers

The following sections describe the outbound and inbound doorbell registers.

12.4.5.1 Outbound Doorbell Register (ODR)

ODR is accessible from the PCI bus and the CSB in both host and agent modes. Figure 12-6 shows the ODR*n* fields.

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 12-7 describes the ODR registers.

Table 12-7. ODR Field Descriptions

Bits	Name	Description
31–29	—	Reserved
28–0	ODR*n*	Outbound doorbell *n.* Write 1 from the CSB to set. Write 1 from the PCI bus to clear. Writing 0 has no effect. (Writing a bit in this register from the CSB causes an interrupt ($\overline{\text{PCI\_INTA}}$) to be generated.)

12.4.5.2 Inbound Doorbell Register (IDR)

IDR is accessible from the PCI bus and the CSB in both host and agent modes. Figure 12-7 shows the IDR fields.

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 12-8 describes the IDR registers.

Table 12-8. IDR Field Descriptions

Bits	Name	Descriptions
31	IMC	Inbound machine check. Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing this bit from the PCI bus causes a machine check interrupt to be generated to the local processor.
30–0	IDR*n*	Inbound doorbell *n*. Write 1 from the PCI bus to set. Write 1 from the CSB to clear. Writing 0 has no effect. Writing a bit in this register from the PCI bus causes an interrupt to be generated to the local processor.

12.4.6 Inbound Message Interrupt Status Register (IMISR)

The IMISR contains the interrupt status of the doorbell and message register events. Writing a 1 to IM1I clears the bit. The events are generated by the PCI masters.

Figure 12-8 shows the IMISR fields.

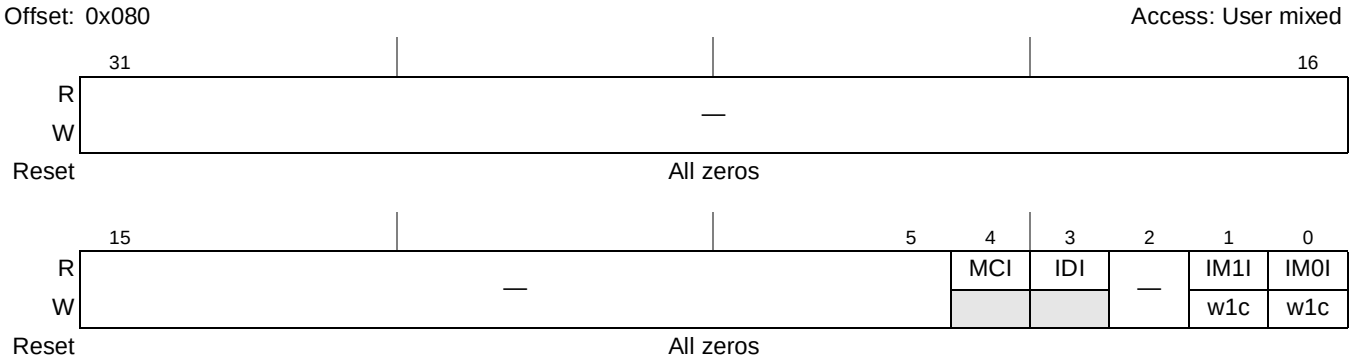


Figure 12-8. Inbound Message Interrupt Status Register (IMISR)

Table 12-9 describes the IMISR register.

Table 12-9. IMISR Field Descriptions

Bits	Name	Descriptions
31–5	—	Reserved
4	MCI	Machine check interrupt. Indicates whether a machine check interrupt condition was generated by setting the IDR[31]. The interrupt is cleared by writing a 1 to IDR[IMC] from the CSB. 0 No machine check interrupt 1 There is a machine check interrupt
3	IDI	Inbound doorbell interrupt. Indicates whether an inbound doorbell interrupt occurred. 0 No inbound doorbell interrupt 1 There is an inbound doorbell interrupt
2	—	Reserved
1	IM1I	Inbound message 1 interrupt. Indicates whether an inbound message 1 interrupt occurred. Write 1 to this position to clear this bit. 0 No Inbound Message 1 interrupt. 1 There is an Inbound Message 1 interrupt.
0	IM0I	Inbound message 0 interrupt. Indicates whether an inbound message 0 interrupt occurred. Write 1 to this position to clear this bit. 0 No inbound message 0 interrupt. 1 There is an inbound message 0 interrupt.

12.4.7 Inbound Message Interrupt Mask Register (IMIMR)

This register contains the interrupt mask of the doorbell and message register events generated by the PCI master. Figure 12-9 shows the IMIMR fields.

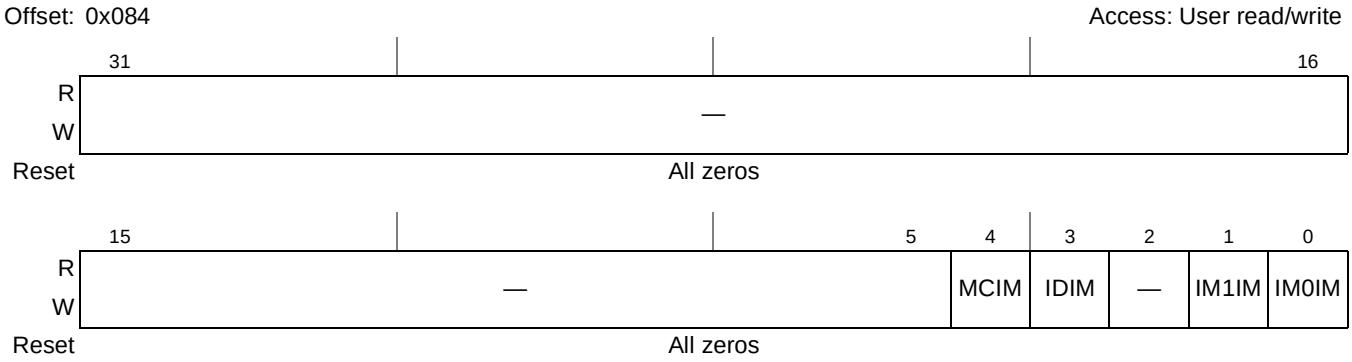


Figure 12-9. Inbound Message Interrupt Mask Register (IMIMR)

Table 12-10 describes the IMISR register.

Table 12-10. IMIMR Field Descriptions

Bits	Name	Description
31–5	—	Reserved
4	MCIM	Machine check interrupt mask. 0 Machine check interrupt from the IDR is allowed 1 Machine check interrupt is masked. IMISR[MC1] is cleared
3	IDIM	Inbound doorbell interrupt mask. 0 Inbound doorbell interrupt is allowed 1 Inbound doorbell interrupt is masked. IMISR[IDI] is cleared.
2	—	Reserved
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

12.4.8 DMA Registers

Each DMA channel has a set of seven 32-bit registers (mode, status, current descriptor address, next descriptor address, source address, destination address, and byte count) to support transactions. The following sections describe the format of the DMA support registers.

12.4.8.1 DMA Mode Register (DMAMR*n*)

This section describes the DMA mode register. The mode register allows software to start the DMA transfer and to control various DMA transfer characteristics. Figure 12-10 shows the DMAMR*n* fields.

Offset: 0x100, 0x180, 0x200, 0x280 Access: User read/write

	31–28	27–24	23–21	20	19	18	17–16
R W	—	DRCNT	BWC	DMSEN	IRQS	EMSEN	DAHTS
Reset	All zeros						

	15–14	13–12	11–10	9–8	7	6–4	3	2	1	0
R W	SAHTS	DAHE	SAHE	PRC	—	EOTIE	—	TEM	CTM	CC	CS
Reset	All zeros									

Figure 12-10. DMA Mode Registers (DMAMR*n*)

Table 12-11 describes the DMAMR*n* register.

Table 12-11. DMAMR*n* Field Descriptions

Bits	Name	Description
31–28	—	Reserved
27–24	DRCNT	DMA request count. This field specifies the number of cache lines transferred per DMA request assertion if EMSEN is 1. This field is not used if EMSEN is 0. 0101 1 cache lineì 0110 2 cache lines 0111 4 cache lines 1000 8 cache lines 1001 16 cache lines 1010 32 cache lines Others 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	EMSEN	External master start enable. This bit is cleared when the DMA transfer has completed, so it must be set again for each transfer. 0 The channel is started by software setting the CS bit 1 The channel is started by hardware asserting the $\overline{\text{DREQ}}$ pin

Table 12-11. 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 address hold feature is not supported when external hardware control is used. **Note:** The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1). **Note:** The DMA does not support address hold for both the source and the destination at the same transfer.
12	SAHE	Source address hold enable. This bit allows the DMA controller to hold the source address constant for every transfer. The size used for transfer is indicated by SAHTS. Note that hardware supports only aligned transfers for this feature. 0 Do not hold the source address constant 1 Hold the source address constant **Note:** The address hold feature is not supported when external hardware control is used. **Note:** The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1). **Note:** The DMA does not support address hold for both the source and the destination at the same transfer.
11–10	PRC	PCI read command. This field indicates the type of PCI read command to use. 00 1 PCI read 01 2 PCI read line 10 4 PCI read multiple 11 8 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 (DMASR*n*[TE] is set) 1 The DMA will complete the transfer regardless of whether a transfer error occurs (the TE bit is not set)

Table 12-11. DMAMR*n* Field Descriptions (continued)

Bits	Name	Description
2	CTM	Channel transfer mode. 0 Chaining mode 1 Direct mode
1	CC	Channel continue. This bit applies only to chaining mode. Setting this bit indicates that the current descriptor segment should be repeated. CC is cleared by the DMA once the repeat takes effect, so it only causes a single repeat. 0 Normal chaining 1 DMACDAR is not loaded from DMANDAR, causing a repeat of the current descriptor segment
0	CS	Channel start. A 0-to-1 transition occurring on this bit when the channel is not busy (SR[CB] bit is 0) will start the DMA process. If the channel is busy and a 0-to-1 transition occurs, the DMA channel will restart from a previous halt condition. A 1-to-0 transition when the channel is busy (CB bit is 1) will halt the DMA process. Nothing happens if the channel is not busy and a 1-to-0 transition occurs. This bit is cleared by the DMA at the end of a transfer.

12.4.8.2 DMA Status Register (DMASR*n*)

This section describes the DMA status register. The status register reports various DMA conditions during and after the DMA transfer. Writing a 1 to a specific set bit clears the bit. Figure 12-11 shows the DMASR*n* fields.

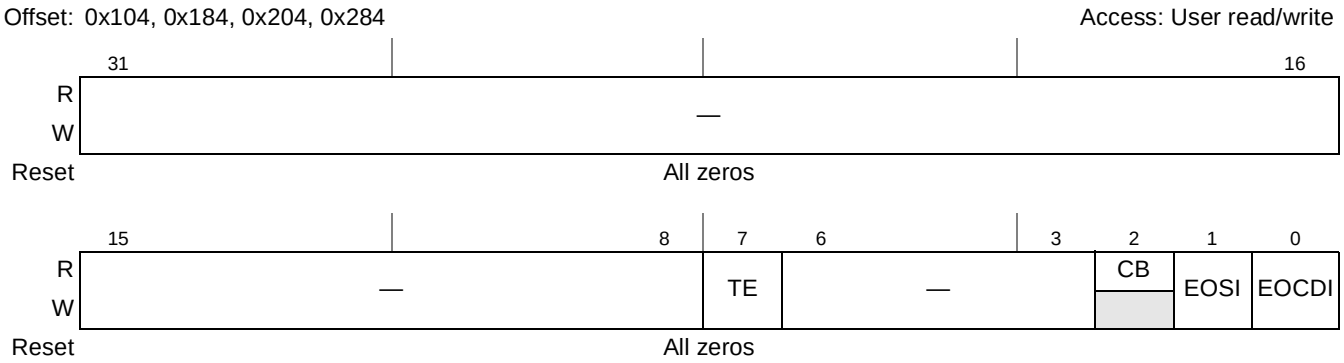


Figure 12-11. DMA Status Register (DMASR*n*)

Table 12-12 describes the DMASR*n* register.

Table 12-12. DMASR*n* Field Descriptions

Bits	Name	Description
31–8	—	Reserved
7	TE	Transfer error. Set when there is an error condition during the DMA transfer and the DMAMR*n*[TEM] bit is cleared.
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: (1) an error, (2) a halt, or (3) completion of the DMA transfer. 0 No DMA transfer is currently in progress 1 A DMA transfer is currently in progress

Table 12-12. 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.

12.4.8.3 DMA Current Descriptor Address Register (DMACDAR*n*)

DMACDAR*n* contains the address of the current segment descriptor being transferred. In chaining mode, software must initialize this register to point to the first descriptor in the chain. After processing the first descriptor, the DMA controller moves the contents of the next descriptor address register into DMACDAR, loads the following descriptor into DMANDAR, and executes the current transfer. Figure 12-12 shows the DMACDAR*n* fields.

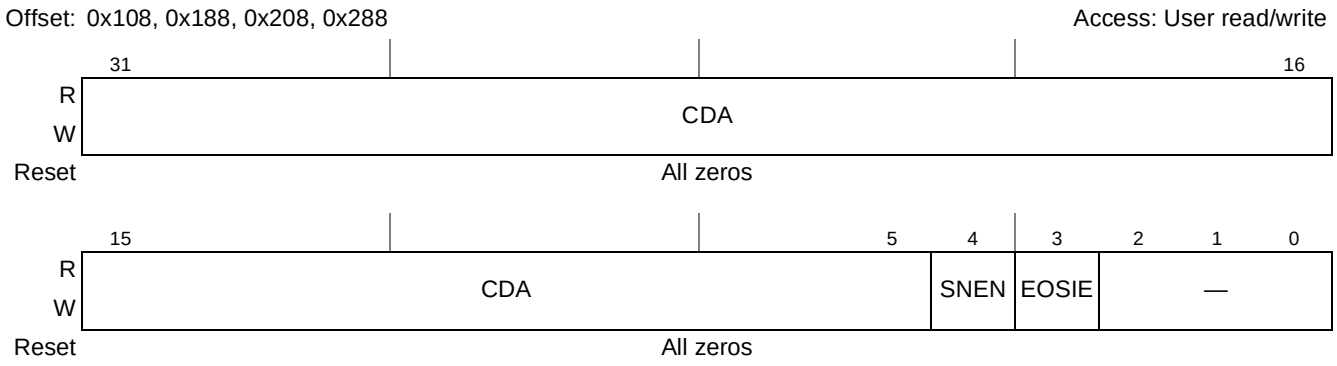


Figure 12-12. DMA Current Descriptor Address Register (DMACDAR*n*)

Table 12-13 describes the DMACDAR*n* register.

Table 12-13. DMACDAR*n* Field Descriptions

Bits	Name	Description
31–5	CDA	Current descriptor address. This field contains the current descriptor address of the segment descriptor in memory. It must be aligned on an 8-word boundary.
4	SNEN	Snoop enable. 0 Snooping is disabled on DMA transactions of the current segment. 1 Snooping is enabled on DMA transactions of the current segment.
3	EOSIE	End-of-segment interrupt enable 0 No end-of-segment interrupt is generated. 1 An interrupt is generated when the current DMA transfer for the current descriptor is finished.
2–0	—	Reserved

12.4.8.4 DMA Source Address Register (DMASAR*n*)

DMASAR*n* indicates the address from which the DMA controller will be reading data. The software must ensure that this is a valid memory address. Figure 12-13 shows the DMASAR*n*.

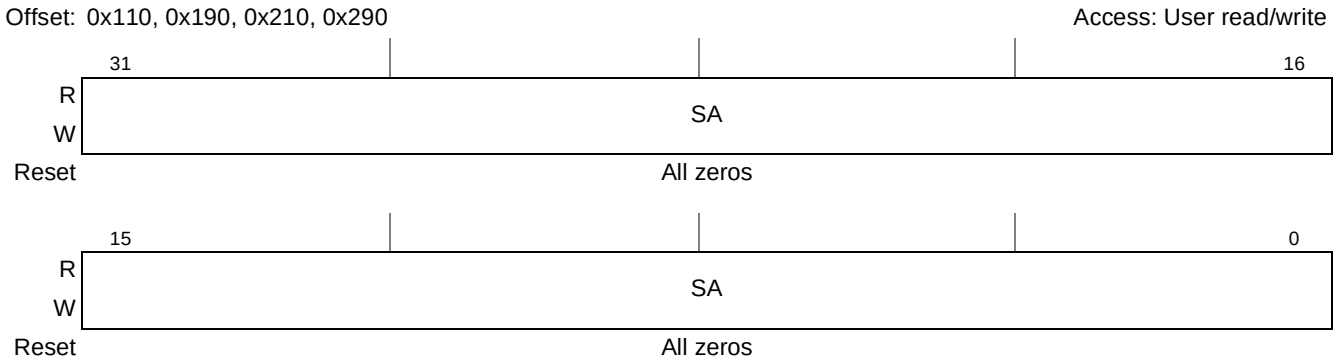


Figure 12-13. DMA Source Address Register (DMASAR*n*)

Table 12-14 describes the DMASAR*n* register.

Table 12-14. DMASAR*n* Field Descriptions

Bits	Name	Description
31–0	SA	Source address of DMA transfer. The content of this field is updated after each DMA read operation.

12.4.8.5 DMA Destination Address Register (DMADAR*n*)

DMADAR*n* indicates the address to which the DMA controller will be writing data. The software must ensure that this is a valid memory address. Figure 12-14 shows the DMADAR*n* fields.

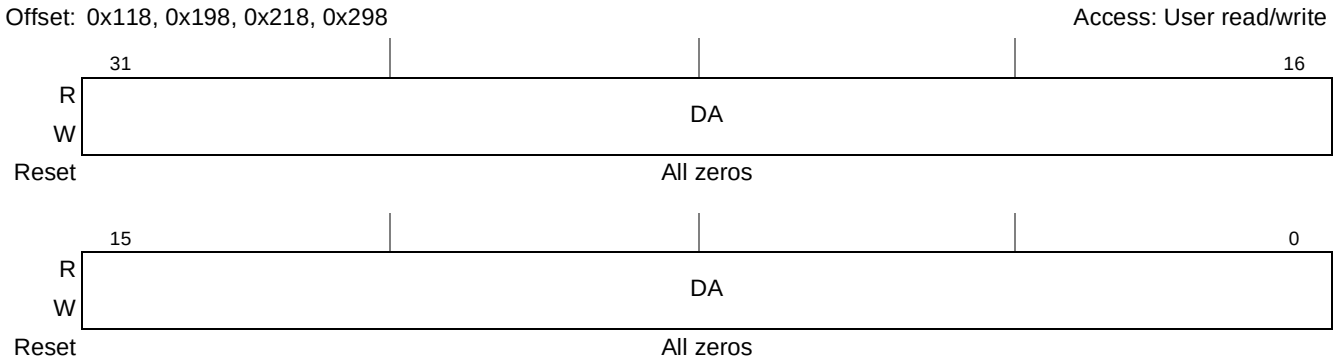


Figure 12-14. DMA Destination Address Register (DMADAR*n*)

Table 12-15 describes the DMADAR*n* register.

Table 12-15. DMASAR*n* Field Descriptions

Bits	Name	Description
31–0	DA	Destination address of DMA transfer.Updated after each DMA write operation.

12.4.8.6 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*.

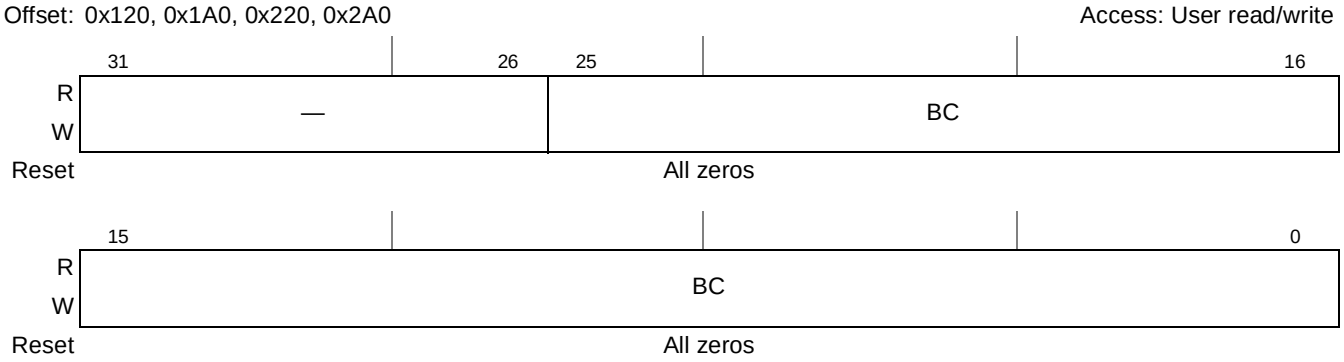


Figure 12-15. DMA Byte Count Register (DMABCR*n*)

Table 12-16 describes the DMABCR*n* register.

Table 12-16. 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.

12.4.8.7 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*.

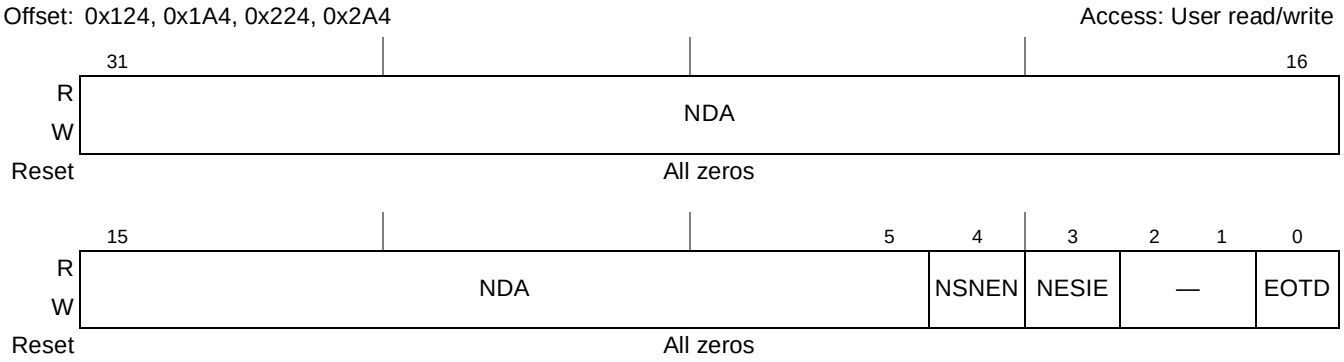


Figure 12-16. DMA Next Descriptor Address Register (DMANDAR*n*)

Table 12-17 describes the DMANDAR*n* register.

Table 12-17. 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.

12.4.8.8 DMA General Status Register (DMAGSR)

DMAGSR provides faster access to the status bits by combining the status bits of all of the DMA channels into one register. Each byte of this register provides the value of bits 7–0 of a channel's DMA status register. These bits are cleared by writing to the individual DMA status registers. Figure 12-17 shows the DMAGSR fields.

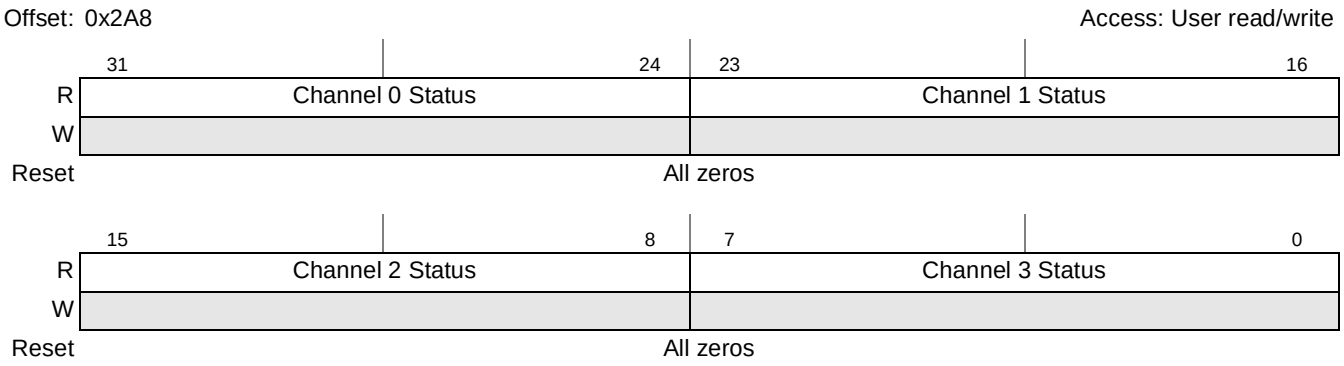


Figure 12-17. DMA General Status Register (DMAGSR)

12.5 Functional Description

12.5.1 Message Unit

An embedded processor is often part of a larger system containing many processors and distributed memory. These processors tend to work on tasks independent of the host and other peripheral processors in the system. Because of the independent nature of the tasks, it is necessary to provide a communication mechanism between the peripheral processors and the rest of the system. One such method is the use of messages. This block provides a messaging unit to further facilitate communications between host and peripheral. The message unit uses generic messages and doorbell registers.

12.5.1.1 Messaging Registers (IMR0–IMR1)

There are two 32-bit inbound message registers (IMR0–IMR1) and two 32-bit outbound message registers (OMR0–OMR1). IMR0 and IMR1allow a remote host or PCI master to write a 32-bit value that in turn causes an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. OMR0 and OMR1 allow the local processor to write an outbound message which, in turn, causes the outbound interrupt signal $\overline{\text{PCI\_INTA}}$ to assert.

The interrupt to the local processor is cleared by writing 1 to the appropriate IMISR bit. The interrupt to PCI ($\overline{\text{PCI\_INTA}}$) is cleared by writing 1 to the appropriate OMISR bit.

12.5.1.2 Doorbell Registers (IDR and ODR)

This block contains the inbound doorbell register (IDR) and the outbound doorbell register (ODR). The inbound doorbell allows a remote processor to set a bit in the register from the PCI bus. This, in turn generates an interrupt request to the on-chip interrupt controller that drives an interrupt line to the local processor. The local processor can write to the ODR, which causes the outbound interrupt signal $\overline{\text{PCI\_INTA}}$ 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.

12.5.2 DMA Controller

The DMA controller transfers blocks of data independent of the local processor or PCI hosts. Data movement occurs on the PCI bus and/or CSB. The DMA module has four high-speed DMA channels, which share buffer space in the IOS to facilitate the gathering and sending of data. Both the local processor and PCI masters can initiate a DMA transfer.

Features of the DMA controller include the following:

- Four channels
- Concurrent execution across multiple channels with programmable bandwidth control
- All channels are accessible by local processor and remote PCI masters
- Unaligned transfer capability
- Data chaining and direct mode
- Interrupt on completed segment, chain, and error

Figure 12-18 shows a diagram of the DMA controller in the integrated device.

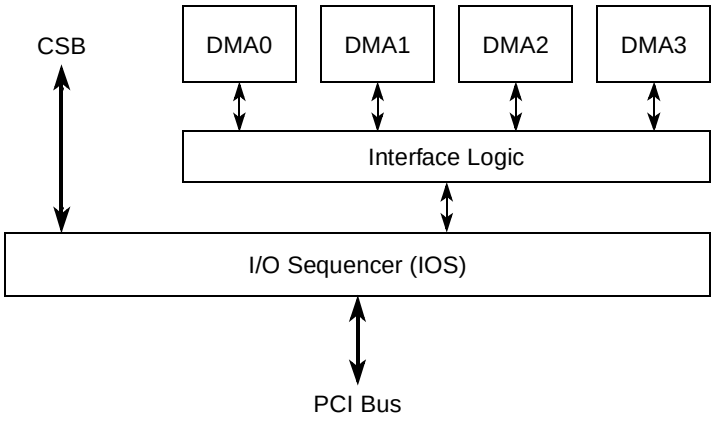


Figure 12-18. DMA Controller Block Diagram

12.5.3 DMA Operation

The DMA controller operates in the following two modes:

- Direct mode, in direct mode, the DMA controller does not read a chain of descriptors from memory but instead uses the current parameters in the DMA registers to start a DMA transfer. The DMA transfer finishes after all the bytes specified in the byte count register have been transferred. See Section 12.6.1, “Initialization Steps in Direct Mode,” for more details on initialization steps.
- Chaining mode, in chaining mode, the DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the descriptor information loaded for each segment. Once the current segment is finished, the DMA controller reads the next descriptor from memory and begins another DMA transfer. The process is finished if the current descriptor is the last one in the chain. See Section 12.6.2, “Initialization Steps in Chaining Mode,” for more details on initialization steps.

In both modes, setting the start bit in the DMA mode register begins the DMA transfer.

The DMA controller supports 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.

12.5.3.1 External Control

The DMA transfer of any channel, in either direct mode or chaining mode, can be controlled by the DMA request input signals. External control is enabled by setting the external master start enable (EMSEN) bit instead of the channel start (CS) bit in the DMA mode register (DMAMR*n*).

When using external control, the following restrictions apply:

- Both the source and destination addresses must be aligned to 32-byte boundaries.
- In chaining mode all byte count values except the last must be multiples of 32 bytes.

A falling edge on $\overline{\text{DREQ}n}$ sets DMAMR*n*[CS] to start the transfer and asserts the corresponding $\overline{\text{DACK}n}$ output signal. The number of cache lines specified by the DMA request count (DRCNT) bit in the DMA mode register (DMAMR*n*)—or the remaining byte count, if smaller— is transferred, and the CS bit and the $\overline{\text{DACK}n}$ output signal are then cleared and negated to halt the transfer until the next $\overline{\text{DREQ}n}$ assertion.

When using the $\overline{\text{DACK}n}$ handshake signal, $\overline{\text{DREQ}n}$ should remain asserted until $\overline{\text{DACK}n}$ is asserted, at which point $\overline{\text{DREQ}n}$ may be negated. $\overline{\text{DREQ}n}$ may be asserted again to resume the transfer or start a new transfer once $\overline{\text{DACK}n}$ has been negated.

If the $\overline{\text{DACK}n}$ handshake signal is not used, $\overline{\text{DREQ}n}$ should remain asserted until the first transaction of the DMA transfer appears on the external interface, at which point $\overline{\text{DREQ}n}$ may be negated. $\overline{\text{DREQ}n}$ may be asserted again to resume the transfer or start a new transfer as early as one clock cycle after its negation, even if the current transfer is still active. The DMA controller is able to record a new assertion of $\overline{\text{DREQ}n}$ while a transfer is in progress, with the effect of setting DMAMR*n*[CS] again once the transfer has been halted.

Once a transfer has been completed, DMAMR*n*[EMSEN] is cleared by the DMA controller, and DMAMR*n*[CS] will not be set again until DMAMR*n*[EMSEN] has been set by the user. The assertion of $\overline{\text{DREQ}n}$ and the setting of DMAMR*n*[EMSEN] may occur in either order; whichever occurs later will trigger the DMA transfer.

The $\overline{\text{DDONE}n}$ output signal is asserted when the DMA transfer has completed, that is, all bytes specified in the byte count register or the descriptors have been transferred. This signal could be used as an indication that the number of cache lines transferred might be smaller than that specified by the DRCNT field.

NOTE

The $\overline{\text{DACK}n}$ and $\overline{\text{DDONE}n}$ output signals are intended as handshake signals for $\overline{\text{DREQ}n}$. They are asserted and negated according to the DMA controller internal logic. These signals are not synchronized to the transactions appearing on the external pins of the device. Specifically, the negation of $\overline{\text{DACK}n}$ or the assertion of $\overline{\text{DDONE}n}$ does not mean that all transactions of the DMA transfer have been completed as seen on the external pins.

See Section 12.6.3, “Initialization Steps in Direct Mode with External Control,” and Section 12.6.4, “Initialization Steps in Chaining Mode with External Control,” for more details on initialization steps.

12.5.3.2 DMA Coherency

The four DMA channels use up to four cache lines (128 bytes) of buffer space in the IOS in addition to 16 bytes of local buffer space. Because no address snooping occurs in these internal queues, data posted in these queues is not visible to the rest of the system while a DMA transfer is in progress. It is the responsibility of application software to ensure the coherency of the region being transferred during the DMA process.

Snooping of the CPU or processor data cache is selectable during DMA transactions. A snoop bit is provided in the DMA current descriptor address register (DMACDAR*n*) and the DMA next descriptor address register (DMANDAR*n*) that allows software to control when the cache is snooped on a per segment basis.

12.5.3.3 Halt and Error Conditions

DMA transfers are halted either by clearing the CS (channel start) bit in the DMA mode register (DMAMR*n*) or when encountering an error condition. In either case, the application software can do one of the following:

- Continue the DMA transfer
- Reconfigure the DMA for a new transfer
- Leave the channel in the halted state

When a DMA channel is halted, its programming model is completely accessible. If the DMA is halted due to an error condition, the TE (transfer error) bit in the DMA status register (DMASR*n*) must be cleared before the transfer can be resumed or a new transfer initiated. Note that the TE bit is not cleared automatically by hardware.

12.5.4 DMA Segment Descriptors

DMA segment descriptors contain the source and destination addresses of the data segment, the segment byte count, and a link to the next descriptor. Segment descriptors are built on cache-line (32-byte) boundaries in either CSB or PCI memory and are linked together into chains using the next-descriptor-address field.

Table 12-18. DMA Segment Descriptor Fields

Descriptor Field	Description
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field will be loaded into the DMA Source Address Register (DMASARn).
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 (DMADARn).
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 (DMANDARn).
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 (DMABCRn).

Application software initializes the current DMA current descriptor address register (DMACDAR*n*) to point to the first descriptor in the chain. For each descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by the descriptor. The DMA controller traverses the descriptor chain until reaching the last descriptor (with its EOTD bit set).

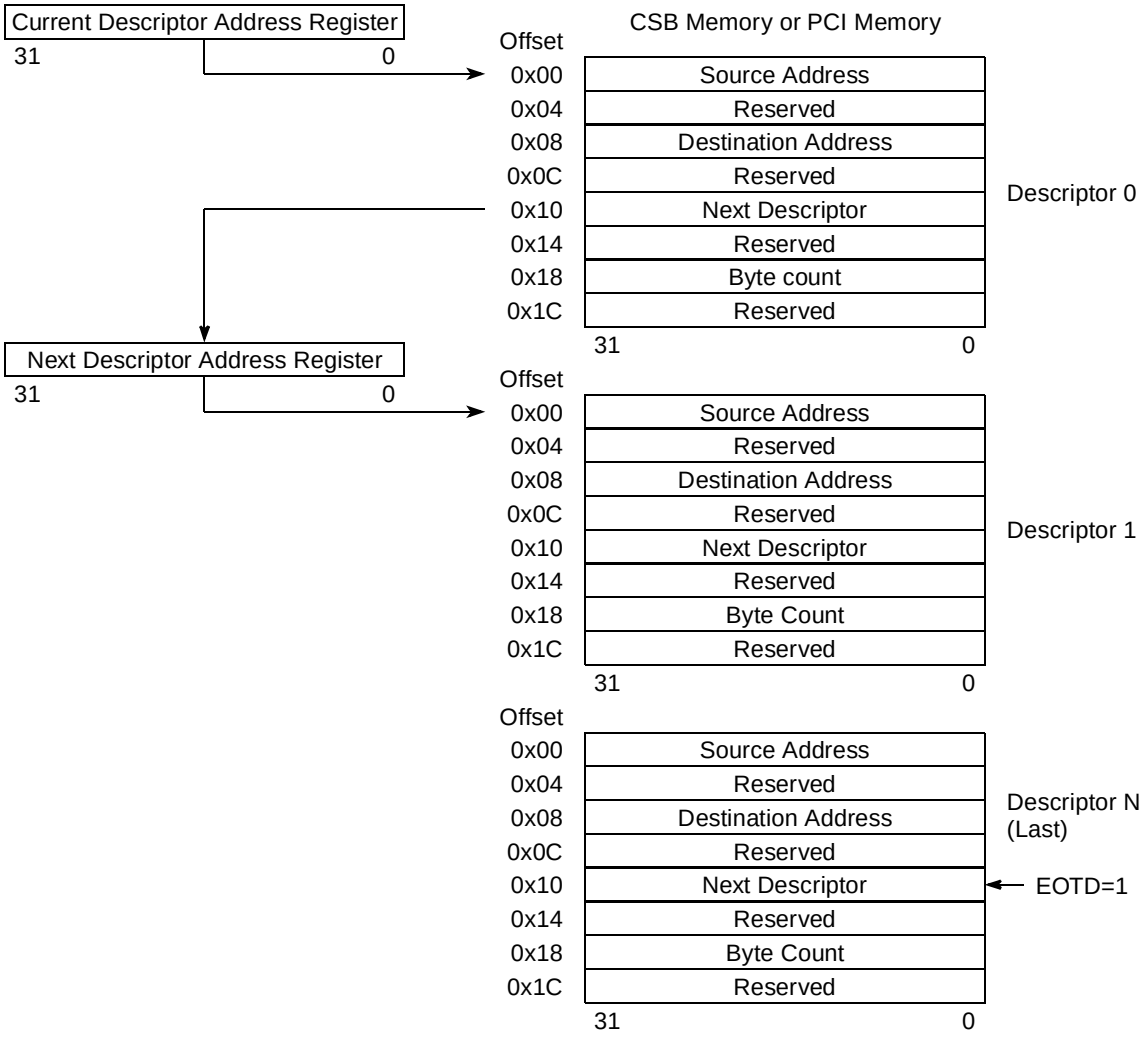


Figure 12-19. DMA Chain of Segment Descriptors

12.5.4.1 Descriptor in Big-Endian Mode

In big-endian mode, the descriptor in CSB memory should be programmed such that data appears in ascending significant-byte order. If segment descriptors are written to memory located in the CSB, they should be treated like they are translated from big-endian to little-endian mode.

Example: Big-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
          double a;           /* 0x1122334455667788 double word*/
          double b;           /* 0x55667788aabbccdd double word*/
          double c;           /* 0x8765432101234567 double word */
          double d;           /* 0x0123456789abcdef double word */
} Descriptor;
Results:  Source Address = 0x44332211 <MSB..LSB>
          Destination Address = 0x88776655 <MSB..LSB>
          Next Descriptor Address = 0x21436587 <MSB..LSB>
          Byte Count = 0x67452301 <MSB..LSB>
```

12.5.4.2 Descriptor in Little-Endian Mode

In little-endian mode, each segment descriptor should be programmed in descending significant-byte order.

Example: Little-endian mode descriptor's data structure. Note that the descriptor structure must be aligned on an 8-word boundary.

```
struct {
          double a;           /* 0x8877665544332211 double word*/
          double b;           /* 0x1122334488776655 double word*/
          double c;           /* 0x7654321012345678 double word */
          double d;           /* 0x0123456776543210 double word */
} Descriptor;
Results:  Source Address = 0x44332211 <MSB..LSB>
          Destination Address = 0x88776655 <MSB..LSB>
          Next Descriptor Address = 0x12345678 <MSB..LSB>
          Byte Count = 0x76543210 <MSB..LSB>
```

12.6 Initialization/Application Information

12.6.1 Initialization Steps in Direct Mode

The initialization steps of a DMA transfer in direct mode are described as follows:

1. Poll the CB (channel busy) bit in the DMA status register (DMASR*n*) to make sure the DMA channel is idle
2. Initialize the DMASAR*n*, DMADAR*n*, and the DMABCR*n*
3. Initialize DMAMR*n*[CTM]) to indicate direct mode. Other control parameters in the mode register can also be initialized here if necessary
4. First clear then set the DMAMR*n*[CS] to start the DMA transfer

12.6.2 Initialization Steps in Chaining Mode

The initialization steps of a DMA transfer in chaining mode are described as follows:

1. Build a chain of descriptor segments in memory. Refer to Section 12.5.4, “DMA Segment Descriptors.”
2. Poll the DMASR*n*[CB] to make sure the DMA channel is idle
3. Initialize the DMACDAR*n* to point to the first descriptor in the chain
4. Initialize the DMAMR*n*[CTM] to indicate chaining mode. Other control parameters in the mode register can also be initialized here if necessary
5. First clear then set the DMAMR*n*[CS] to start the DMA transfer

12.6.3 Initialization Steps in Direct Mode with External Control

The initialization steps of a DMA transfer in direct mode with external control are described as follows:

1. Poll the CB (channel busy) bit in the DMA status register (DMASR*n*) to make sure the DMA channel is idle.
2. Initialize the DMA source address register (DMASAR*n*), the DMA destination address register (DMADAR*n*), and the DMA byte count register (DMABCR*n*).
3. Set DMAMR*n*[CTM]) to indicate direct mode, program the DRCNT field, and set the EMSEN bit. Other control parameters in the mode register can also be initialized here if necessary.

12.6.4 Initialization Steps in Chaining Mode with External Control

The initialization steps of a DMA transfer in chaining mode with external control are described as follows:

1. Build a chain of descriptor segments in memory. Refer to Section 12.5.4, “DMA Segment Descriptors,” for more information.
2. Poll the CB (channel busy) bit in the DMA status register (DMASR*n*) to make sure the DMA channel is idle.
3. Initialize the DMA current descriptor address register (DMACDAR*n*) to point to the first descriptor in the chain.
4. Clear the DMAMR*n*[CTM] to indicate chaining mode, program the DRCNT field, and set the EMSEN bit. Other control parameters in the mode register can also be initialized here if necessary.

Chapter 13
PCI Bus Interface

The PCI interface is compatible with the *PCI Local Bus Specification*, Rev. 2.3. It is beyond the scope of this manual to document the intricacies of PCI. This chapter describes the PCI controller and provides a basic description of the PCI bus operations. The specific emphasis is directed at how this device implements the PCI specification. Designers of systems incorporating PCI devices should refer to the respective specifications for a thorough description of the PCI buses.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Because this is inconsistent with the terminology in this manual, the terms ‘word’ and ‘double word’ are not used in this chapter. Instead, the number of bits or bytes indicates the exact quantity.

13.1 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.

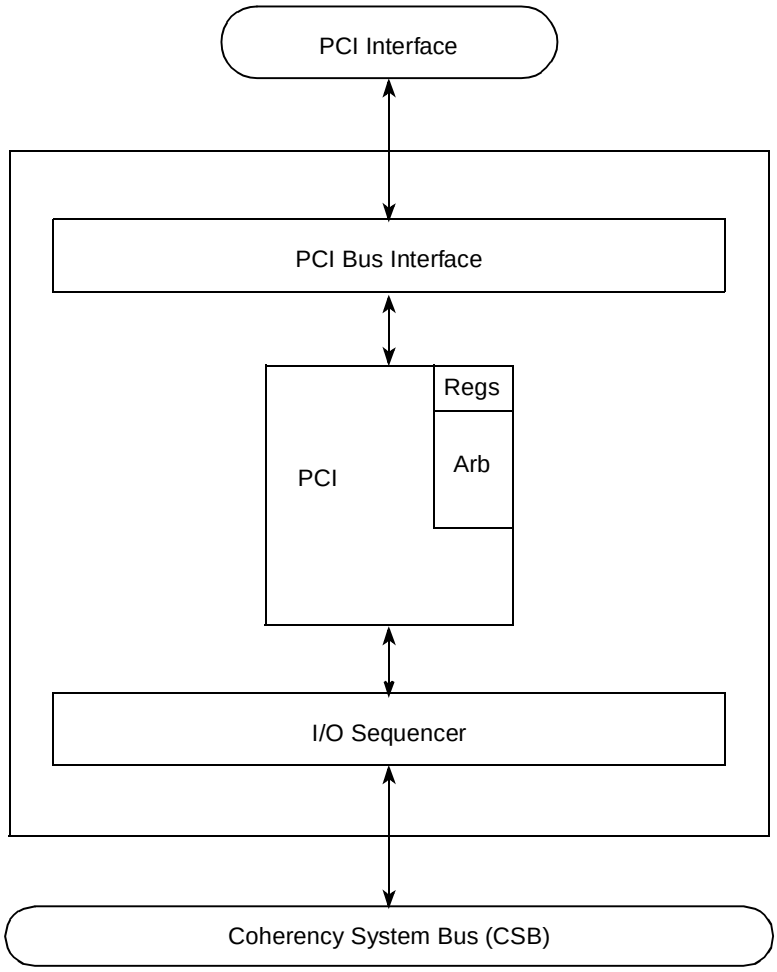


Figure 13-1. PCI Controller Block Diagram

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.	4.3.2.2.1/4-17
PCI clock output enable	Enables the PCI clock output signals.	/4-32
PCI arbiter enable	Enables the on-chip PCI bus arbiter	4.3.2.2/4-16

13.1.2.1 PCI Enable Configuration

The $\overline{\text{PCI\_MODE}}$ signal enables PCI mode for QUICC Engine parallel I/O ports, which can function as PCI signals. See Section 5.4.1.1, “PCI_MODE Signal,” for more information.

13.1.2.2 Host/Agent Mode Configuration

The PCI controller can function as either a PCI host bridge (referred to as host mode) or a peripheral device on the PCI bus (referred to as agent mode). Note that host/agent mode selection is determined at power-up as summarized 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.

13.1.2.3 PCI Clock Output Enable Configuration

Normally when the device is in host mode, PCI clock outputs are provided for connection to other agents in the system. However it is possible to disable the PCI clock outputs of the device. When PCI clock outputs are disabled, PCI clock distribution should be done on the board by an external device. The benefit of disabling the PCI clock outputs is that the PCI clock device pins can be used for alternate functions.

13.1.2.4 PCI Arbiter Configuration

The interface can be configured to use an on-chip or off-chip PCI arbiter. Arbitration for PCI is determined by the value in RCWH[PCIARB]. 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	—
PCI_DEVSEL	PCI device select	High impedance	Required
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	—	—

Table 13-2. Signal Properties (continued)

Name	Function	Reset State	Pull Up
PCI_INTA	PCI interrupt A	High impedance	Required
PCI_IRDY	PCI initiator ready	High impedance	Required
PCI_PAR	PCI parity	High impedance	—
PCI_PERR	PCI parity error	High impedance	Required
PCI_RESET_OUT	PCI reset output	Asserted	
PCI_SERR	PCI system error	High impedance	Required
PCI_STOP	PCI stop	High impedance	Required
PCI_TRDY	PCI target ready	High impedance	Required

Figure 13-2 shows the external PCI signals.

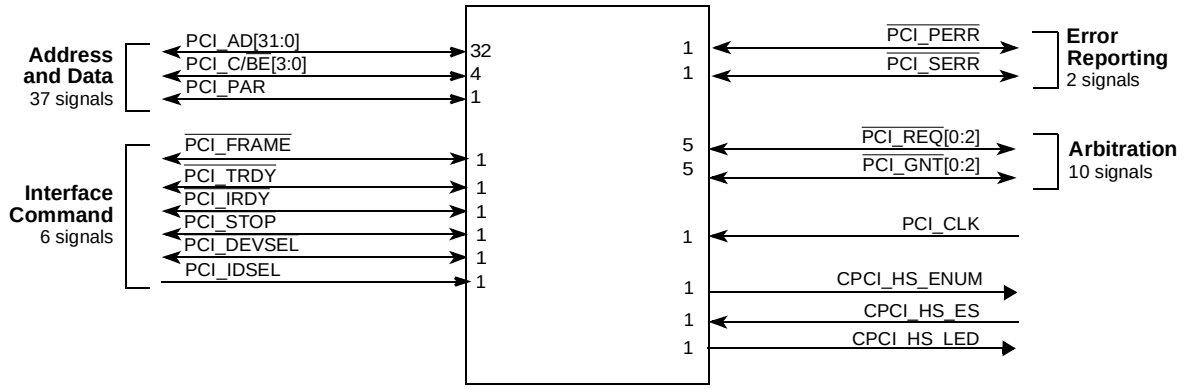


Figure 13-2. PCI Interface External Signals

Table 13-3 contains detailed descriptions of the external PCI interface signals.

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
CPCI_HS_ENUM	O	CompactPCI hot swap enumerator. Used for the hot swap interface to connect to the host as the enumeration request in a compact PCI system. This signal is used for agent mode only.	
		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

Table 13-3. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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
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 Description (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	
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 PCI_TRDY to indicate that valid data is present on PCI_AD[31:0]. During a write, this PCI controller asserts 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 PCI_TRDY to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates 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	Use	Access	Section/Page
PCI Configuration Access Registers			
0x0	PCI_CONFIG_ADDRESS	W	13.3.1.1/13-13
0x4	PCI_CONFIG_DATA	R/W	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	Use	Access	Section/Page
PCI Error Management Registers			
0x00	PCI error status register (PCI_ESR)	w1c	13.3.2.1/13-15
0x04	PCI error capture disable register (PCI_ECDR)	R/W	13.3.2.2/13-16
0x08	PCI error enable register (PCI_EER)	R/W	13.3.2.3/13-17
0x0C	PCI error attributes capture register (PCI_EATCR)	R/W	13.3.2.4/13-18
0x10	PCI error address capture register (PCI_EACR)	R	13.3.2.5/13-19
0x14	PCI error extended address capture register (PCI_EEACR)	R	13.3.2.6/13-20
0x18	PCI error data capture register (PCI_EDCR)	R/W	13.3.2.7/13-20
0x851C	Reserved	—	—
PCI Control and Status Registers			
0x20	PCI general control register (PCI_GCR)	R/W	13.3.2.8/13-21
0x24	PCI error control register (PCI_ECR)	R/W	13.3.2.9/13-21
0x28	PCI general status register (PCI_GSR)	R	13.3.2.10/13-22
PCI Inbound ATU Registers			
0x38	PCI inbound translation address register 2 (PITAR2)	R/W	13.3.2.11/13-23
0x3C	Reserved	—	—
0x40	PCI inbound base address register 2 (PIBAR2)	R/W	13.3.2.12/13-23
0x44	PCI inbound extended base address register 2 (PIEBAR2)	R/W	13.3.2.13/13-24
0x48	PCI inbound window attributes register 2 (PIWAR2)	R/W	13.3.2.14/13-24
0x50	PCI inbound translation address register 1 (PITAR1)	R/W	13.3.2.11/13-23
0x54	Reserved	—	—
0x58	PCI inbound base address register 1 (PIBAR1)	R/W	13.3.2.12/13-23
0x5C	PCI inbound extended base address register 1 (PIEBAR1)	R/W	13.3.2.13/13-24
0x60	PCI inbound window attributes register 1 (PIWAR1)	R/W	13.3.2.14/13-24
0x68	PCI inbound translation address register 0 (PITAR0)	R/W	13.3.2.11/13-23
0x6C	Reserved	—	—
0x70	PCI inbound base address register 0 (PIBAR0)	R/W	13.3.2.12/13-23
0x78	PCI inbound window attributes register 0 (PIWAR0)	R/W	13.3.2.13/13-24
0x7C–0xFF	Reserved	—	—

13.3.1 PCI Configuration Access Registers

This section describes the registers used to allow a local bus master to access the PCI configuration space, and generate special cycle or interrupt acknowledge transactions on the PCI bus. A special case provides access to the PCI controller's internal PCI configuration registers. The PCI registers, PCI_CONFIG_ADDRESS, PCI_CONFIG_DATA, and PCI_INT_ACK, are little endian registers.

13.3.1.1 PCI_CONFIG_ADDRESS

Figure 13-3 shows the PCI_CONFIG_ADDRESS register fields.

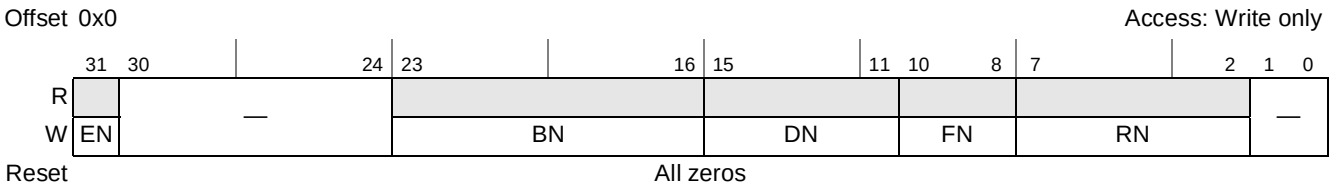


Figure 13-3. PCI_CONFIG_ADDRESS Register

The PCI_CONFIG_ADDRESS register holds the address for an access to the PCI configuration space from the local bus. This register must be programmed before accessing PCI_CONFIG_DATA to perform the transaction. Only 32-bit accesses are permitted.

If EN=1, BN=0, and DN=0, the access is to the internal PCI configuration registers, so no transaction is generated on the PCI bus.

If EN=1, BN=0, DN=31, FN=7, and RN=0, writing to PCI_CONFIG_DATA generates a special cycle transaction and reading from PCI_CONFIG_DATA generates an interrupt acknowledge transaction.

Table 13-6 shows the bit settings of the PCI_CONFIG_ADDRESS register.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions

Bits	Name	Description
31	EN	Enable configuration transaction. Determines the type of transaction to be generated. 0 No configuration transaction will be generated by accessing the CONFIG_DATA register. Such an access will be passed through to the PCI bus as an I/O transaction. Since this is generally not desirable, the user should not access CONFIG_DATA when the EN bit is 0. 1 A configuration transaction will be generated by accessing the CONFIG_DATA register if BN and DN are not both zero.
30–24	—	Reserved
23–16	BN	Bus number. Specifies the bus segment to which a configuration transaction is directed. If this field is 0, a Type 0 configuration transaction is generated. Otherwise, a Type 1 configuration transaction is generated.

Table 13-6. PCI_CONFIG_ADDRESS Field Descriptions (continued)

Bits	Name	Description
15–11	DN	Device number. Specifies the device to which a configuration transaction is directed. For a Type 0 configuration transaction, this field is decoded to individual PCI1_IDSEL signals for the address phase according to the following values. For a Type 1 configuration transaction, this field is used directly for the address phase. **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.

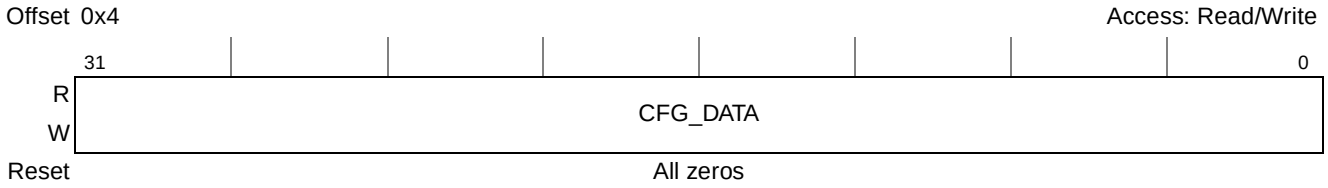


Figure 13-4. PCI_CONFIG_DATA

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.

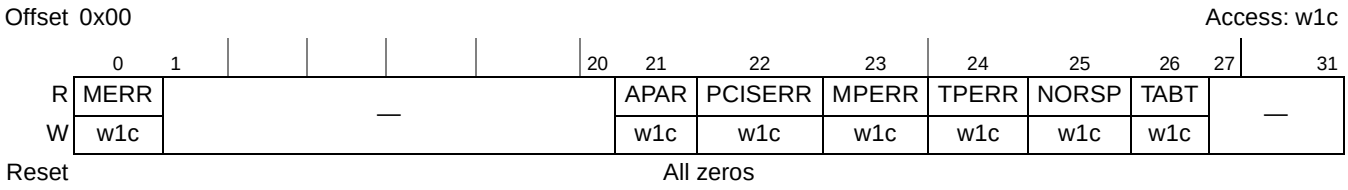


Figure 13-5. PCI Error Status Register (PCI_ESR)

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.

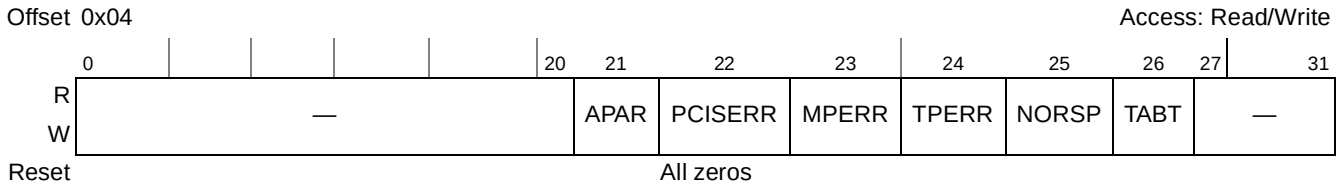


Figure 13-6. PCI Error Capture Disable Register (PCI_ECDR)

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.

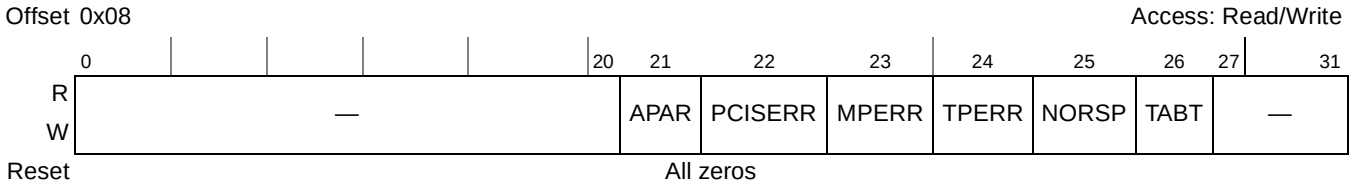


Figure 13-7. PCI Error Enable Register (PCI_EER)

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.

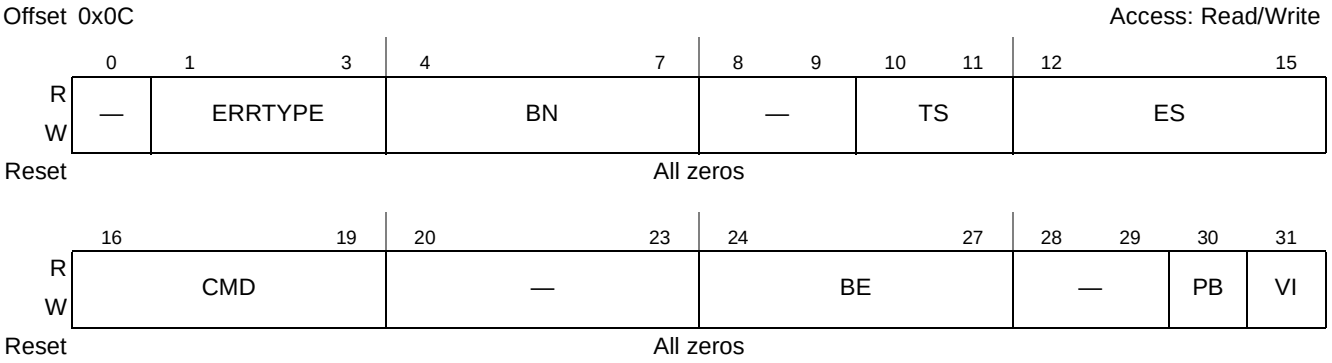


Figure 13-8. PCI Error Attributes Capture Register (PCI_EATCR)

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.

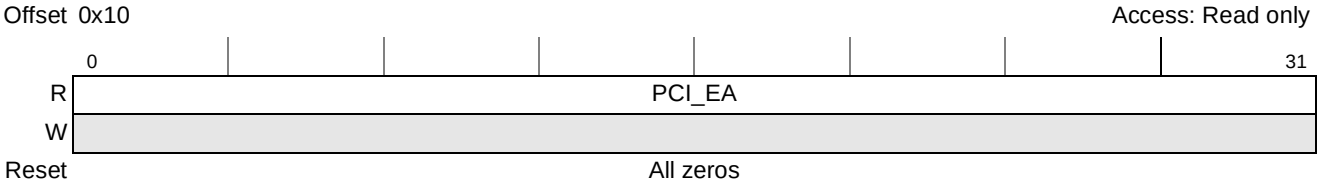


Figure 13-9. PCI Error Address Capture Register (PCI_EACR)

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.

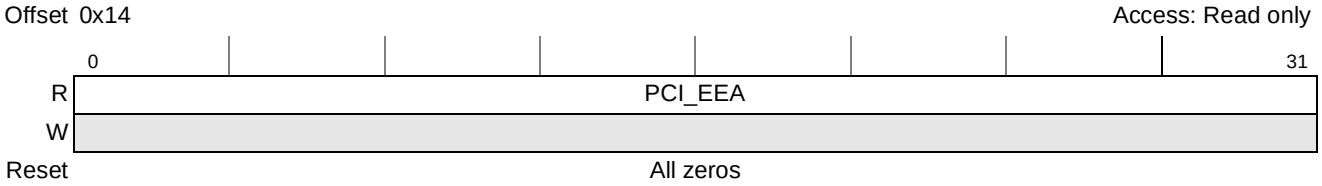


Figure 13-10. PCI Error Extended Address Capture Register (PCI_EEACR)

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.

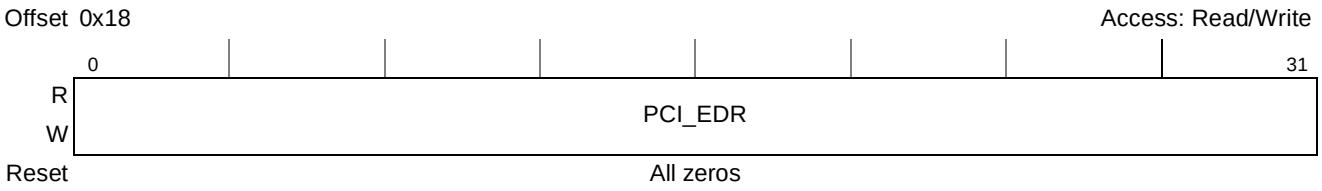


Figure 13-11. PCI Error Data Low Capture Register (PCI_EDLCR)

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.

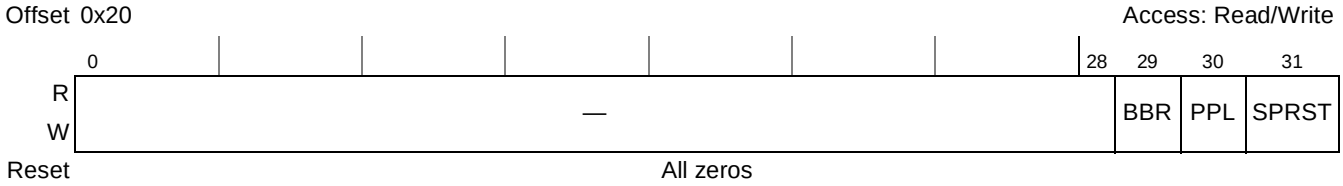


Figure 13-12. PCI General Control Register (PCI_GCR)

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.

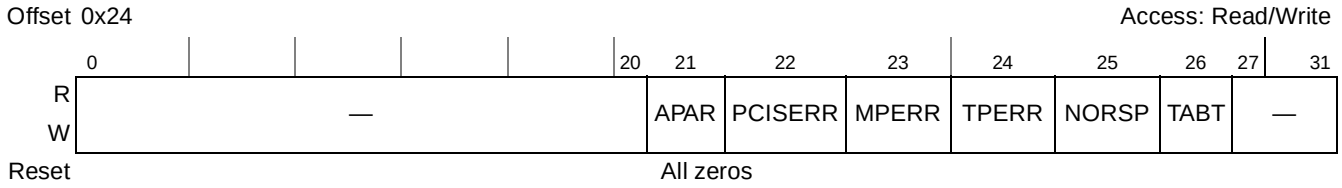


Figure 13-13. PCI Error Control Register (PCI_ECR)

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.

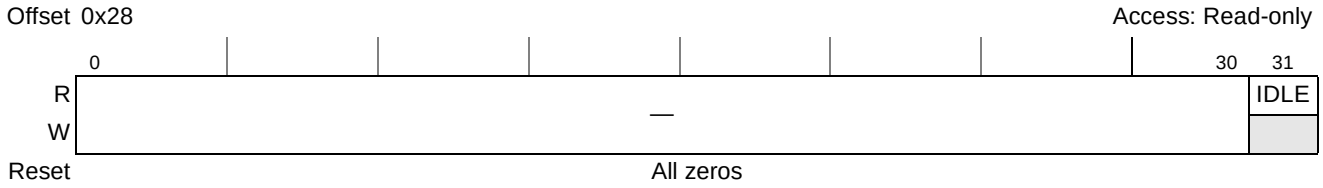


Figure 13-14. PCI General Status Register (PCI_GSR)

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 Chapter 11, “Sequencer for more information on outbound address translation registers.

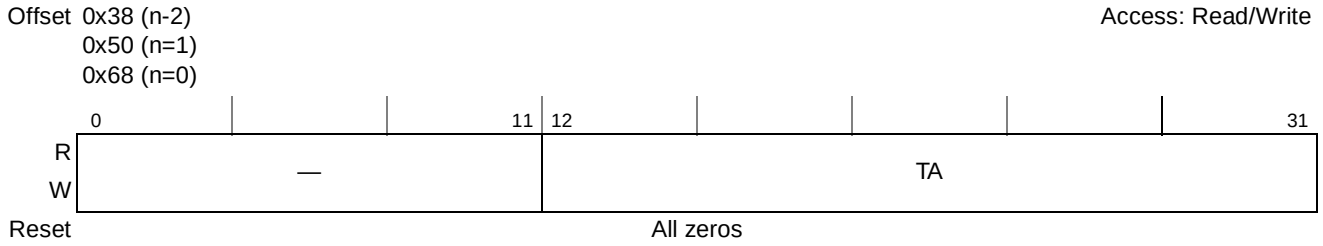


Figure 13-15. PCI Inbound Translation Address Registers (PITAR*n*)

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.

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.

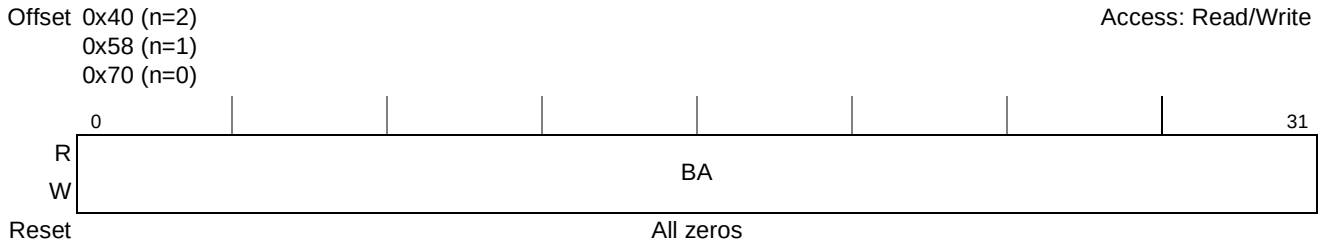


Figure 13-16. PCI Inbound Base Address Registers (PIBAR*n*)

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.

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.

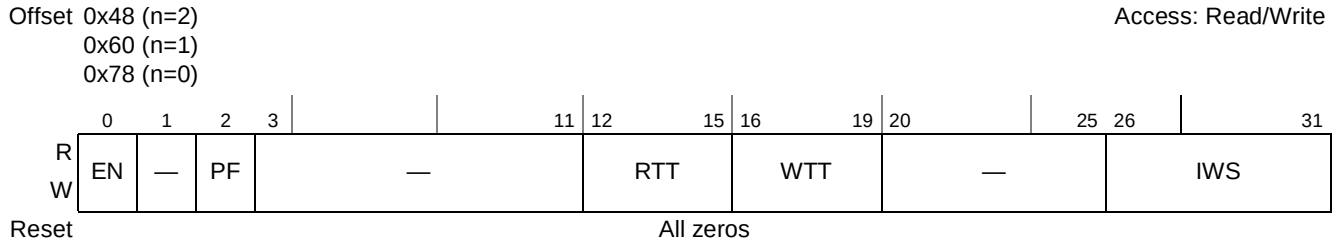


Figure 13-18. PCI Inbound Window Attribute Registers (PIWAR*n*)

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
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) 000000–001010Reserved 0010114-Kbyte window size 0011008-Kbyte window size ... 0111102-Gbyte window size 011111–111111Reserved

13.3.3 PCI Configuration Space Registers

This section describes the PCI configuration space registers. These registers are shown with descending bit numbering to correspond to the PCI standard.

NOTE

The registers described in this section use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data. No byte swapping occurs when the registers are accessed from the PCI bus.

Table 13-22 shows the PCI configuration registers that are mapped in PCI configuration space. Some fields are common to registers in both spaces to ensure consistency. These fields are discussed in the register definitions.

Table 13-22. PCI Configuration Space Registers

Address	Use	Access
00	Vendor ID configuration register	R
02	Device ID configuration register	R
04	PCI command configuration register	R/W
06	PCI status configuration register	Read/bit-reset
08	Revision ID configuration register	R
09	Standard programming interface	R
0A	Subclass code configuration register	R
0B	Base class code configuration register	R
0C	Cache line size configuration register	R/W
0D	Latency timer configuration register	R/W
0E	Header type configuration register	R
0F	BIST control configuration register	R
10	PIMMR base address register	R/W
14	GPL base address register 0	R/W
18	GPL base address register 1	R/W
1C	GPL extended base address register 1	R/W
20	GPL base address register 2	R/W
24	GPL extended base address register 2	R/W
2C	Subsystem vendor ID configuration register	R
2E	Subsystem device ID configuration register	R
34	Capabilities pointer configuration register	R
3C	Interrupt line configuration register	R/W
3D	Interrupt pin configuration register	R
3E	Minimum grant configuration register	R
3F	Maximum latency configuration register	R
44	PCI function configuration register	R/W
46	PCI arbiter control register (PCIACR)	R/W
48	Hot swap register block	R/W

13.3.3.1 Vendor ID Configuration Register

Figure 13-19 shows the vendor ID fields. This is a read-only register.

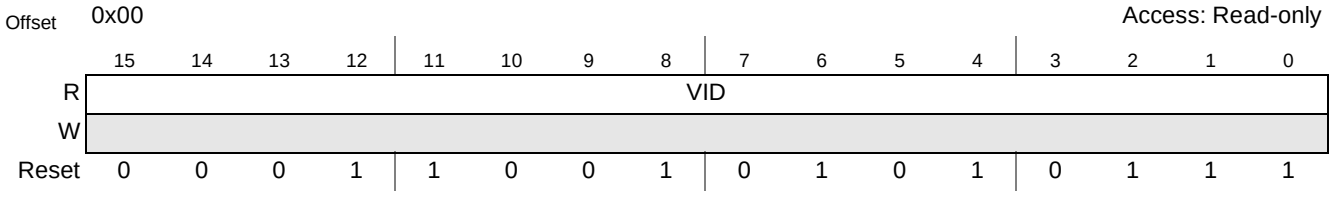


Figure 13-19. Vendor ID Configuration Register

Table 13-23 shows the bit settings of the vendor ID register.

Table 13-23. Vendor ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	VID	Vendor ID. The read-only value 0x1957 specifies Freescale Semiconductor as the manufacturer of the device.

13.3.3.2 Device ID Configuration Register

Figure 13-20 shows the device ID fields. This is a read only register.

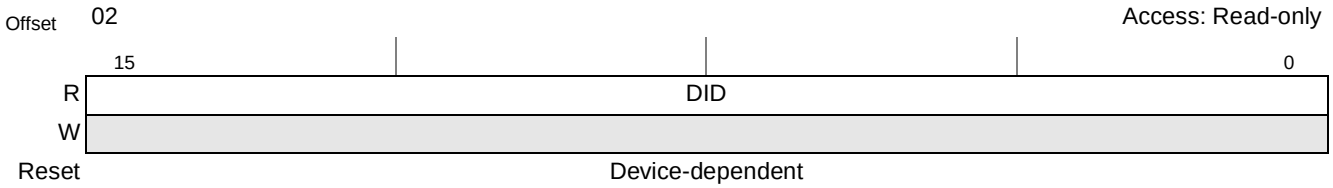


Figure 13-20. Device ID Configuration Register

Table 13-24 shows the bit settings of the device ID register.

Table 13-24. Device ID Configuration Register Field Descriptions

Bits	Name	Description
15–0	DID	Device ID. This field identifies the device. 0091 MPC8360 TBGA 0090 MPC8360E TBGA 0093 MPC8358 TBGA 0092 MPC8358E TBGA 0097 MPC8358 PBGA 0096 MPC8358E PBGA

13.3.3.3 PCI Command Configuration Register

Figure 13-21 shows the PCI command fields.

Offset 04 Access: Mixed

	15	11	10	9	8	7	6	5	4	3	2	1	0
R	—		INTD	FB-B	SERREN	—	PERRR	—	MWI	SC	BMST	MEM	I/O
W													
Reset	All zeros												

Figure 13-21. PCI Command Configuration Register

Table 13-25 shows the bit settings of the PCI command register.

Table 13-25. PCI Command Configuration Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	INTD	Interrupt Disable. Setting this bit masks the $\overline{\text{PCI\_INTA}}$ output. 0 $\overline{\text{PCI\_INTA}}$ provides the device interrupt status. 1 $\overline{\text{PCI\_INTA}}$ is always negated.
9	FB-B	Fast back-to-back. Hard-wired to 0.
8	SERREN	SERR enable. This bit is an enable bit for the SERR driver. Address parity errors are reported only if this bit and bit 6 are 1. 0 $\overline{\text{PCI\_SERR}}$ is never asserted. 1 $\overline{\text{PCI\_SERR}}$ may be asserted to indicate error conditions.
7	—	Reserved
6	PERRR	Parity error response. Controls the PCI controller's response to a parity error. 0 Parity errors are ignored and normal operation continues. 1 Standard parity error treatment.
5	—	Reserved
4	MWI	Memory-write-and-invalidate. Hard-wired to 0.
3	SC	Special cycles. Hard-wired to 0.
2	BMST	Bus master. Controls the PCI controller's ability to be a master on the PCI bus. At reset, this bit is cleared in Agent Mode and set in Host Mode. 0 The PCI controller does not generate PCI accesses. 1 The PCI controller behaves as a bus master.
1	MEM	Memory space. Controls the response to memory space accesses. 0 The PCI controller does not respond to Memory Space accesses. 1 The PCI controller as a target responds to Memory Space accesses.
0	I/O	I/O space. Hard-wired to 0.

13.3.3.4 PCI Status Configuration Register

This register is used to record status information for PCI bus-related events. Some of the bits are hard-wired to indicate the capabilities of the PCI controller. Other bits can be cleared by writing 1 to the bit location. Figure 13-22 shows the PCI status fields.

Offset 06 Access: Mixed

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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.
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.

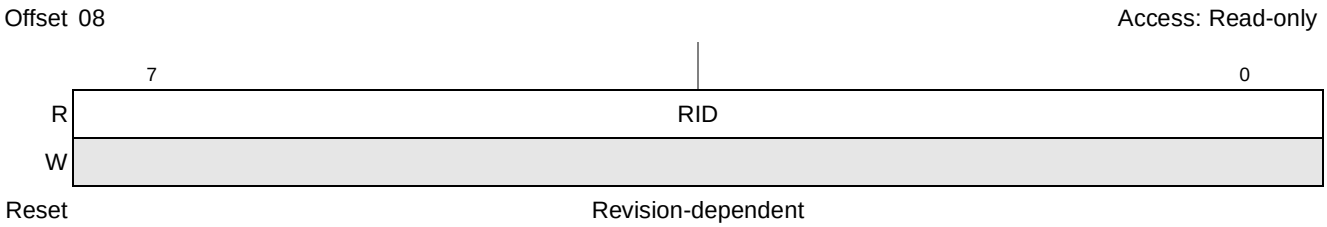


Figure 13-23. Revision ID Configuration Register

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.

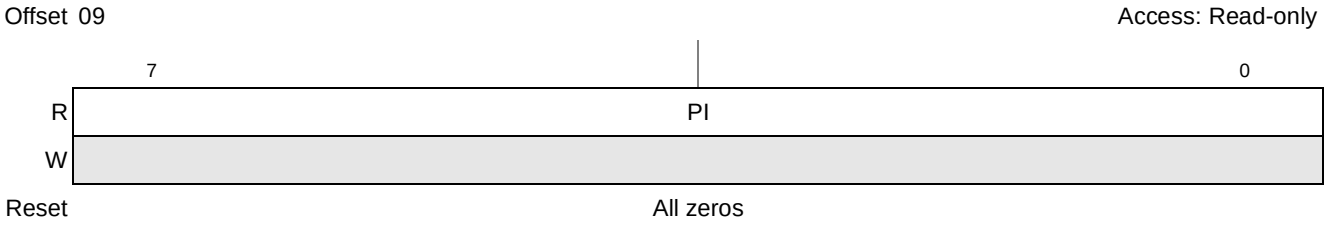


Figure 13-24. Standard Programming Interface Configuration Register

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.

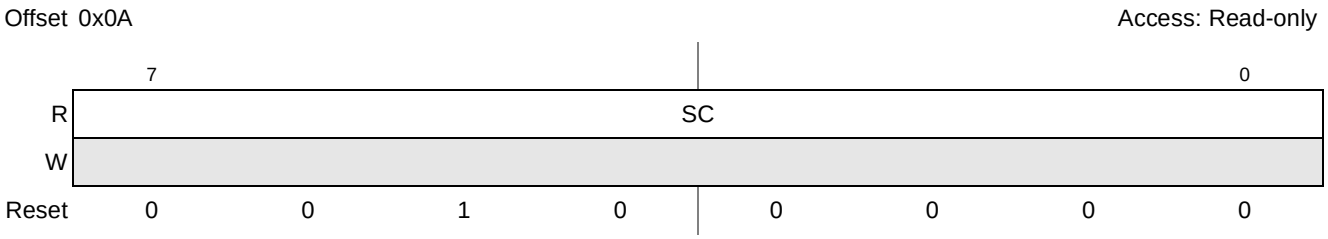


Figure 13-25. Subclass Code Configuration Register

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.

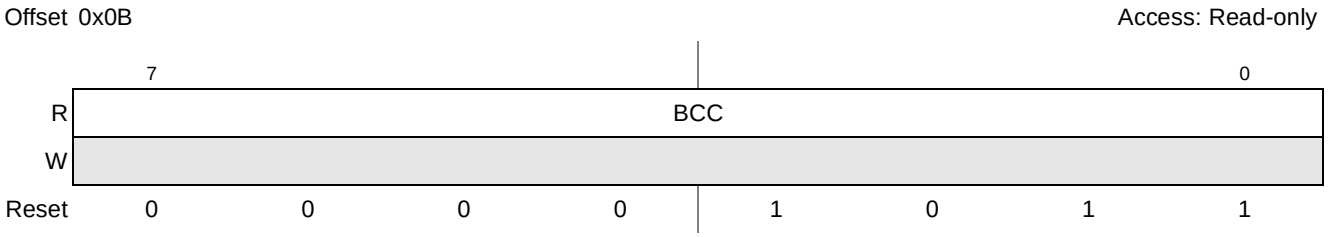


Figure 13-26. Base Class Code Configuration Register

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.

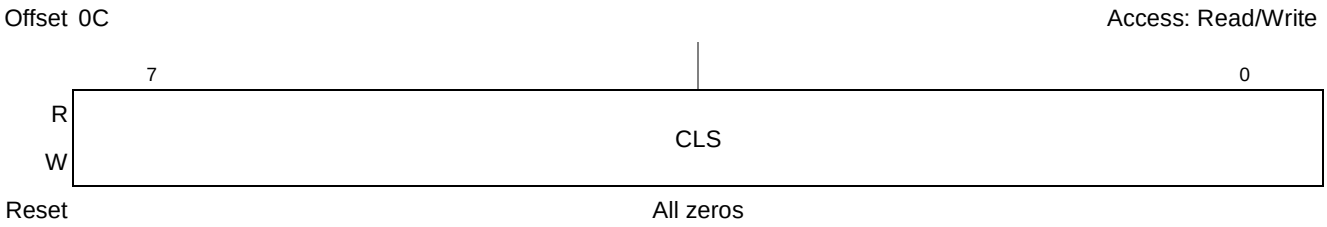


Figure 13-27. Cache Line Size Configuration Register

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.

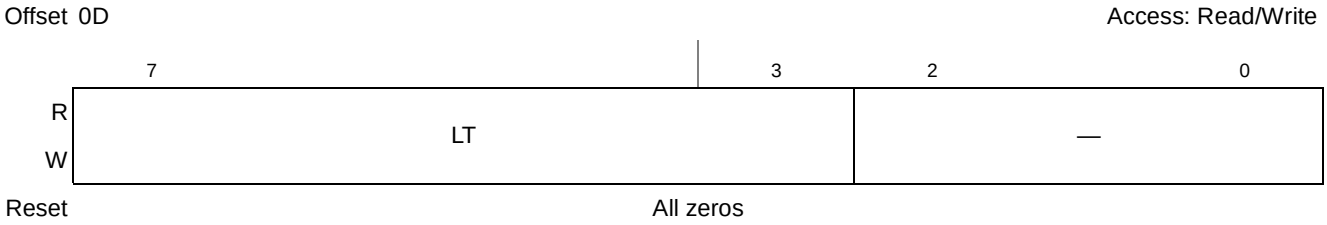


Figure 13-28. Latency Timer Configuration Register

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.

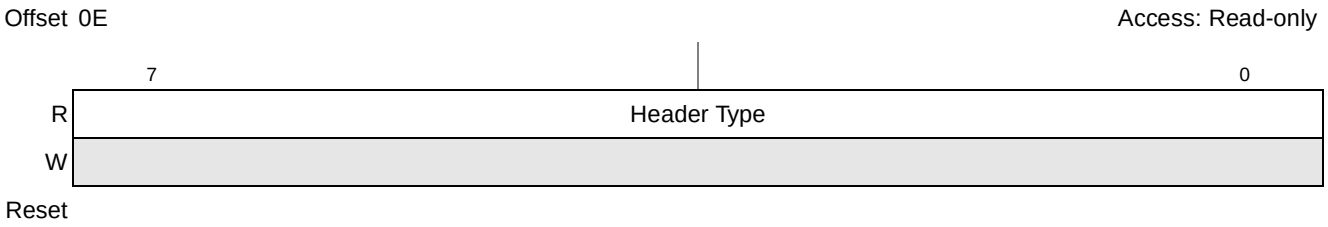


Figure 13-29. Header Type Configuration Register

13.3.3.12 BIST Control Configuration Register

Figure 13-30 shows the read-only BIST control register, which is hard-wired to 0x00.

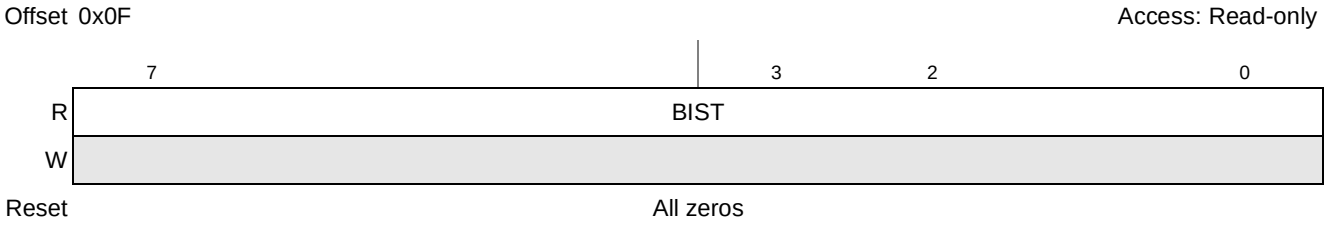


Figure 13-30. BIST Control Configuration Register

13.3.3.13 PIMMR Base Address Configuration Register

Figure 13-31 shows the PIMMR base address register fields.

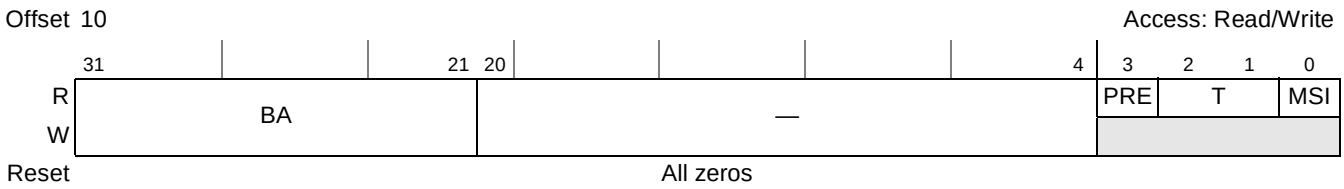


Figure 13-31. PIMMR Base Address Configuration Register

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.

Table 13-33. PIMMR Base Address Configuration Register Field Descriptions (continued)

Bits	Name	Description
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.14 GPL Base Address Register 0

The GPL base address register 0 is provided to allow access to local memory space. This register is closely tied to PIBAR0 and PIWAR0 in the CSR memory space. A write to GPL base address register 0 also causes a change in the base address bits that are not masked according to the IWS field of PIWAR0 in PIBAR0. Note that this write operation will not change the bits that are masked by the IWS field. For read operation these masked bits will always return zeros.

Figure 13-32 shows the GPL base address register 0 fields.

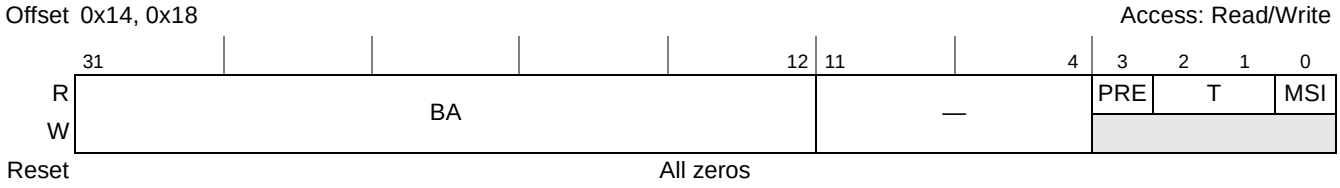


Figure 13-32. GPL Base Address Register 0

Table 13-34 shows the bit settings of the GPL base address register 0.

Table 13-34. GPL Base Address Register 0 Field Descriptions

Bits	Name	Description
31–12	BA	Base address. Defines the base address for the inbound window. Bits 11–4 are hard-wired to 0 since the minimum window size is 4 Kbytes.
3	PRE	Prefetchable. This bit is read-only and contains the value of the PF bit in PIWAR0.
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

13.3.3.15 GPL Base Address Registers 1–2

The general purpose local access base address registers are provided to allow access to local memory space. These registers are closely tied to PIBAR*n* and PIWAR*n* in the CSR memory space. A write to a GPL base address register also causes a change in the base address bits that are not masked according to the IWS field of PIWAR*n* in the corresponding PIBAR*n*. Note that this write operation will not change the bits that are masked by the IWS field. For read operations, these masked bits always return zeros. Figure 13-33 shows the GPL base address register 1–2 fields.

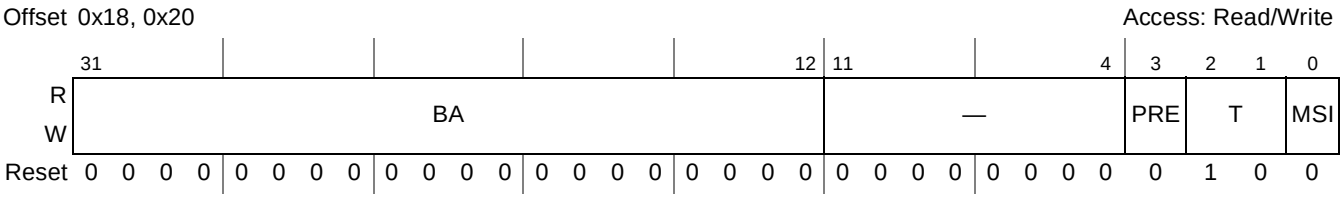


Figure 13-33. GPL Base Address Registers 1–2

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 since 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.

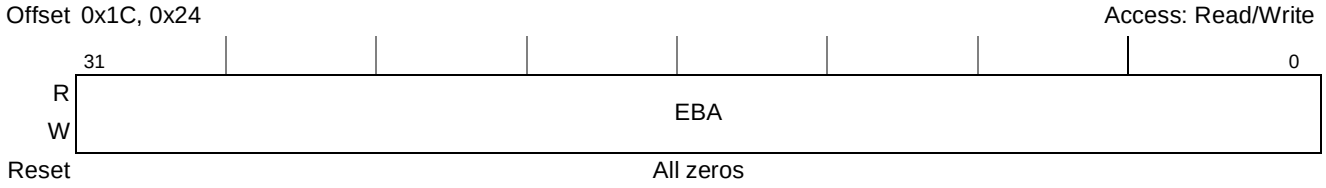


Figure 13-34. GPL Extended Base Address Registers 1–2

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.

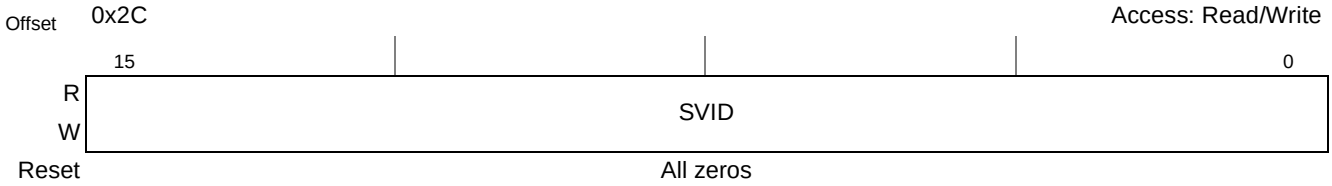


Figure 13-35. Subsystem Vendor ID Configuration Register

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.

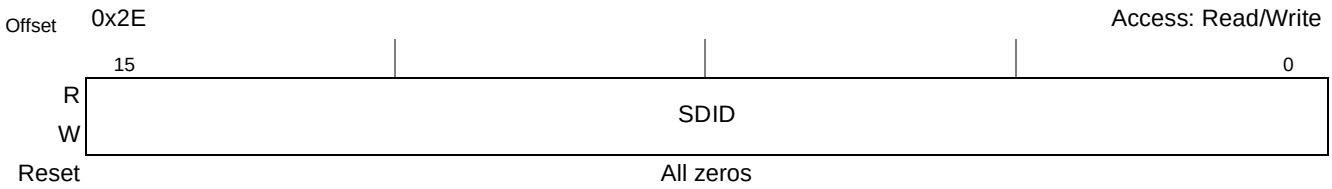


Figure 13-36. Subsystem Device ID Configuration Register

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.

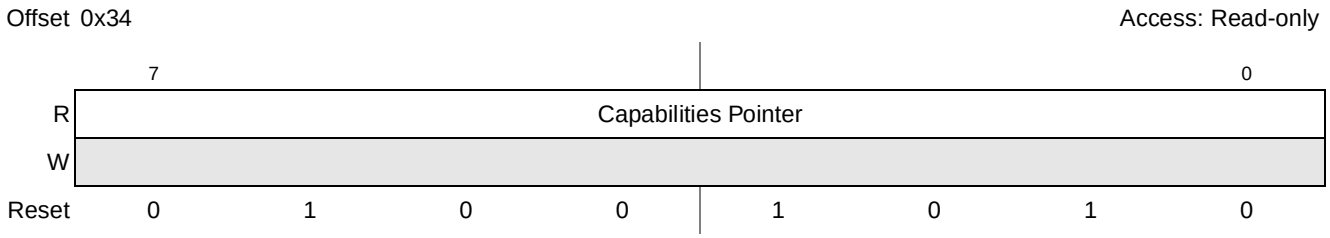


Figure 13-37. Capabilities Pointer Configuration Register

13.3.3.20 Interrupt Line Configuration Register

Figure 13-38 shows the interrupt line configuration register fields.

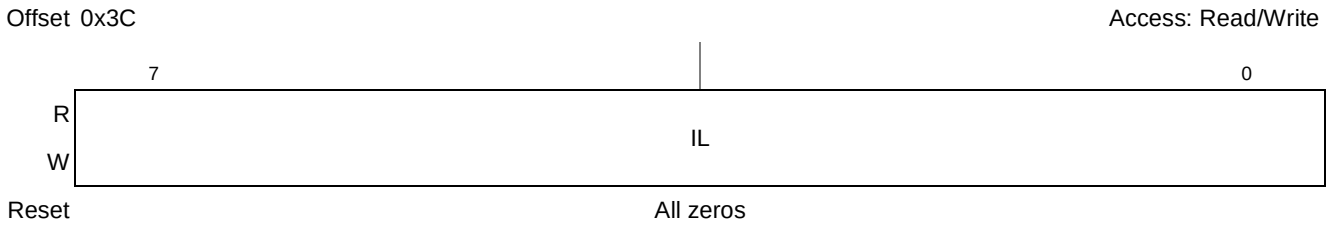


Figure 13-38. Interrupt Line Configuration Register

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.

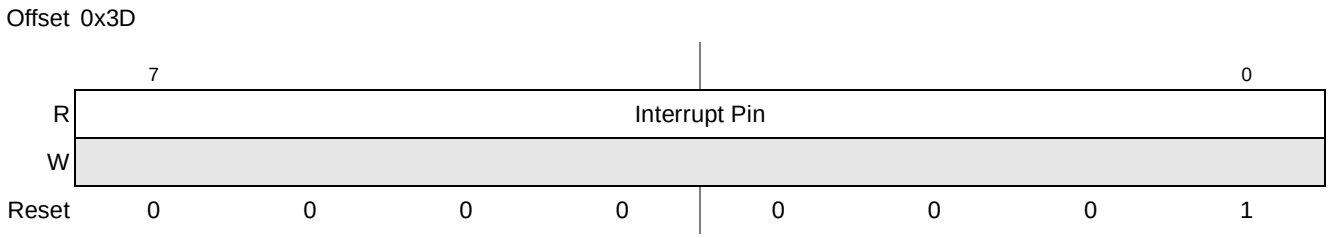


Figure 13-39. Interrupt Pin Register

13.3.3.22 Minimum Grant Configuration Register

Figure 13-40 shows the minimum grant configuration register fields.

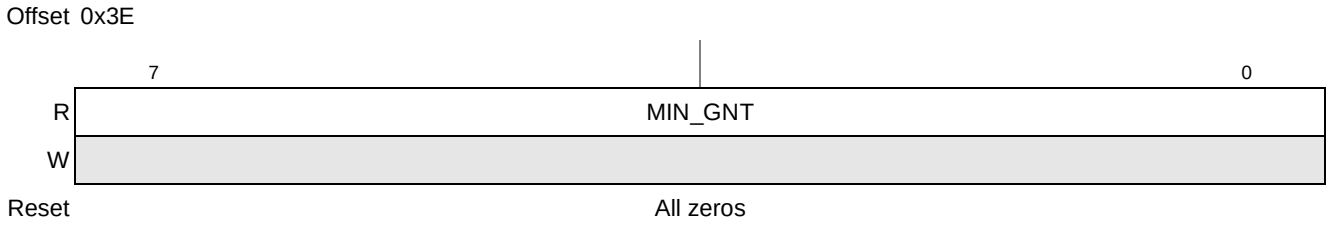


Figure 13-40. Minimum Grant Configuration Register

13.3.3.23 Maximum Latency Configuration Register

Figure 13-41 shows the maximum latency configuration register fields.

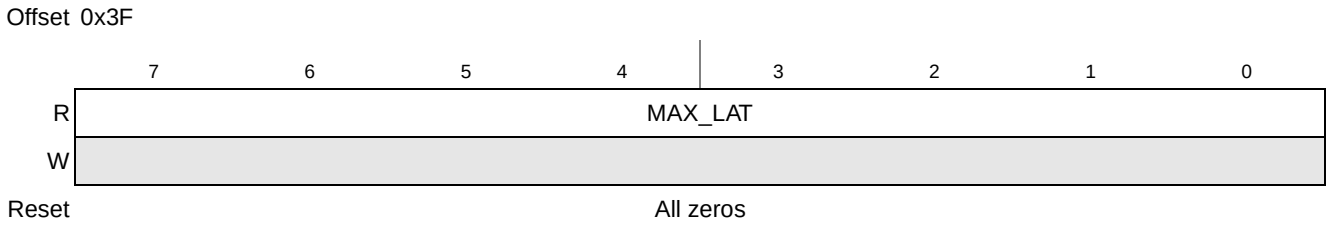


Figure 13-41. Maximum Latency Configuration Register

13.3.3.24 PCI Function Configuration Register

Figure 13-42 shows the PCI function configuration register fields.

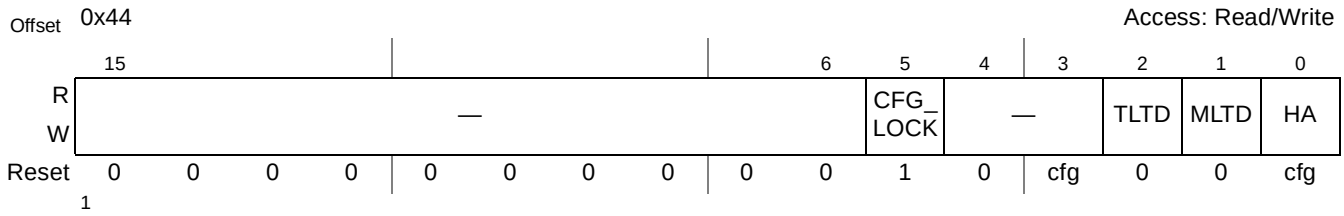


Figure 13-42. PCI Function Configuration Register

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

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.

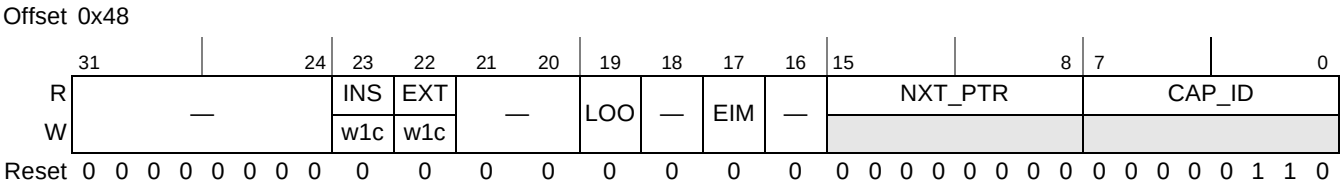


Figure 13-44. Hot Swap Register Block

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.

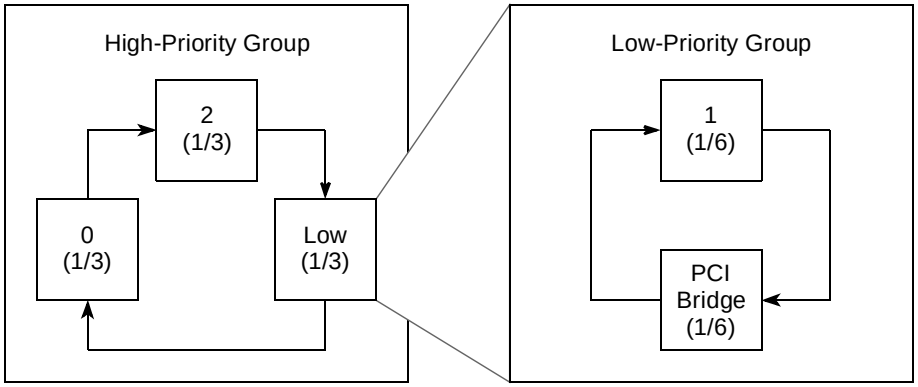


Figure 13-45. PCI Arbitration Example

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/$\overline{\text{BE}}$[3:0] during the address phase of the transaction. PCI bus commands are described in Table 13-43.

Table 13-43. PCI Command Definitions

PCI_C/$\overline{\text{BE}}$[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.
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.
- Three-stated signals with slashes between the two rails have indeterminate values.
- The terms ‘edge’ and ‘clock edge’ refer to the rising edge of the clock.
- The terms ‘asserted’ and ‘negated’ refer to the globally visible state of the signal on the clock edge, and not to signal transitions.
- The symbol  represents a turnaround-cycle.

13.4.3.7 Read and Write Transactions

Both read and write transactions begin with an address phase followed by a data phase. The address phase occurs when $\overline{\text{PCI\_FRAME}}$ is asserted for the first time, and the AD[31:0] signals contain a byte address and the PCI_C/$\overline{\text{BE}}$[3:0] signals contain a bus command. The data phase consists of the actual data transfer and possible wait cycles; the byte enable signals remain actively driven from the first clock of the data phase through the end of the data transfer.

A read transaction starts when $\overline{\text{PCI\_FRAME}}$ is asserted for the first time and the PCI_C/$\overline{\text{BE}}$[3:0] signals indicate a read command. Figure 13-46 shows an example of a single beat read transaction.

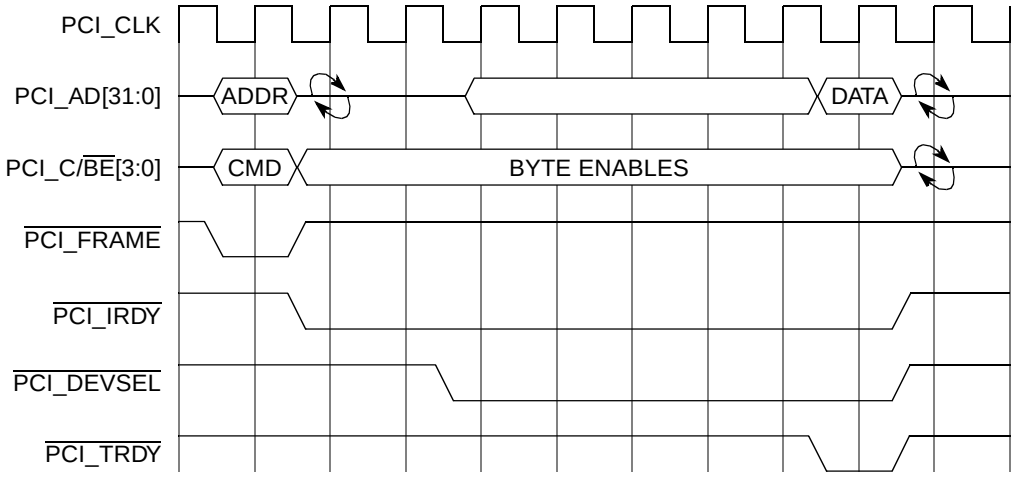


Figure 13-46. Single Beat Read Example

Figure 13-47 shows an example of a burst read transaction.

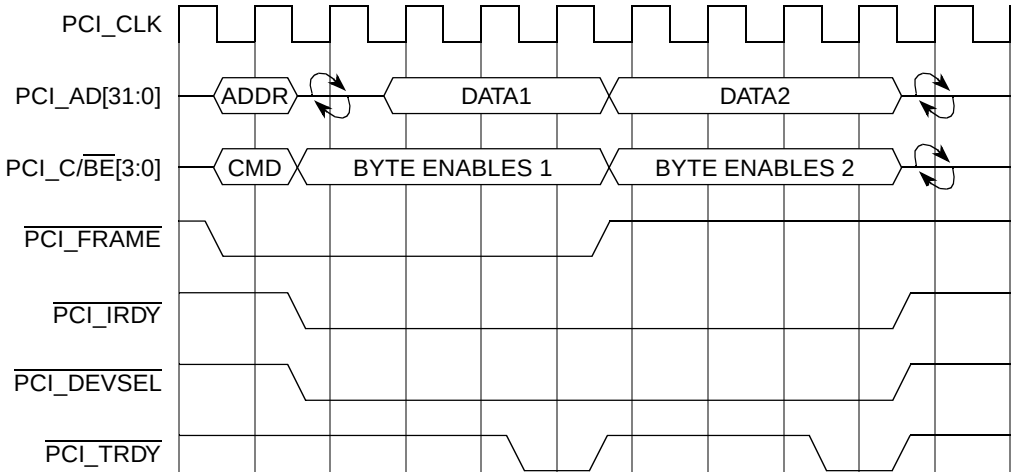


Figure 13-47. Burst Read Example

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.

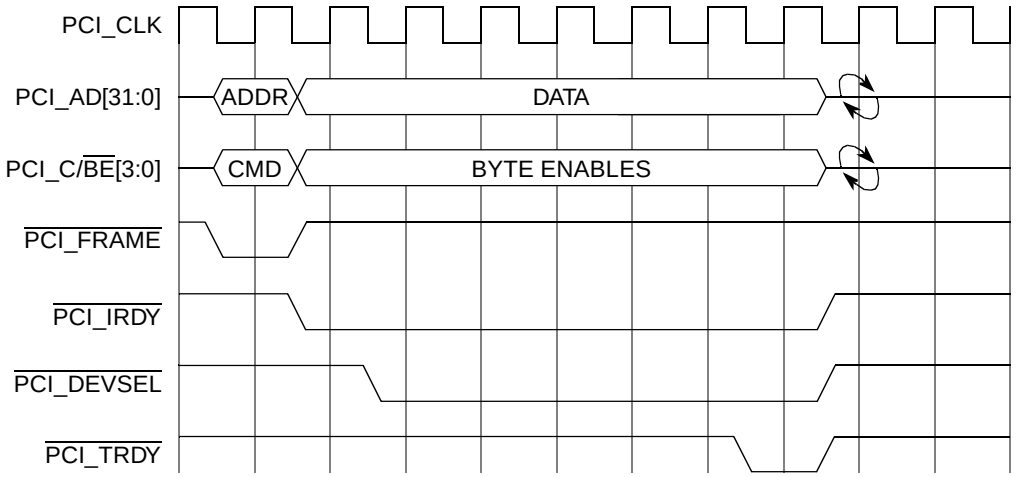


Figure 13-48. Single Beat Write Example

Figure 13-49 shows an example of a burst write transaction.

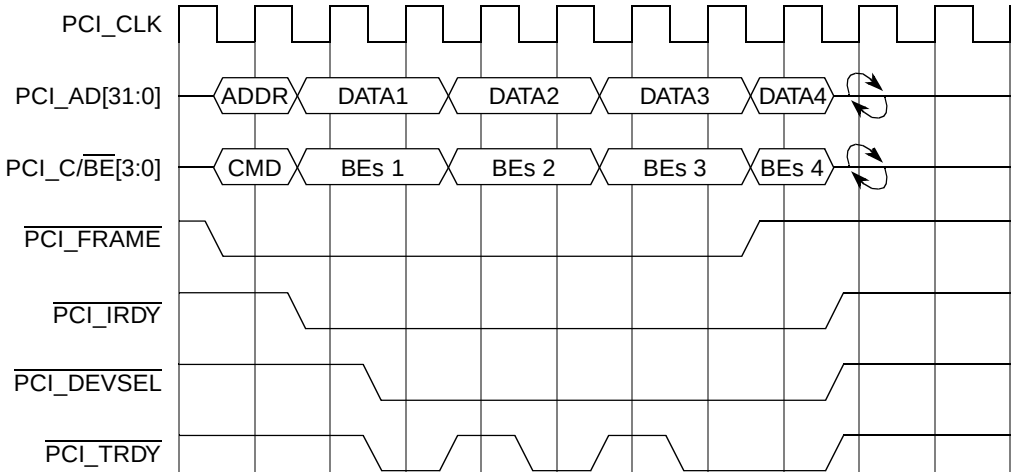


Figure 13-49. Burst Write Example

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).

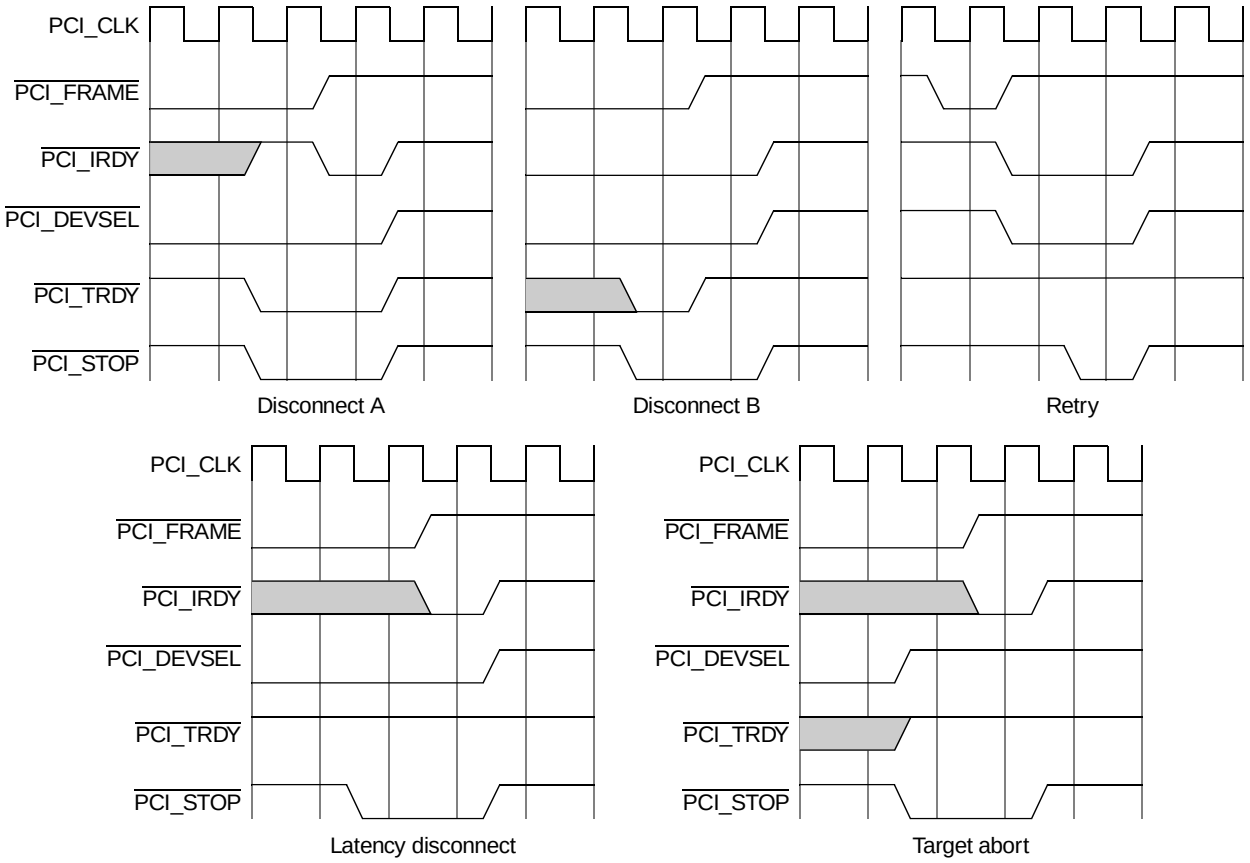


Figure 13-50. Target-Initiated Terminations

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 $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted), if it wants to do only one more data transfer and then stop, it may assert $\overline{\text{PCI\_TRDY}}$ and $\overline{\text{PCI\_STOP}}$ at the same time.

Target-abort refers to the abnormal termination that is used when a fatal error has occurred, or when a target will never be able to respond. Target-abort is indicated when $\overline{\text{PCI\_STOP}}$ is asserted and $\overline{\text{PCI\_DEVSEL}}$ is negated. This indicates that the target requires the transaction to be terminated and does not want the transaction tried again. Note that any transferred data may have been corrupted.

The PCI controller terminates a transaction with target-abort in the case in which it is the intended target of a read transaction from system memory and the data from memory is corrupt. If the PCI controller is the intended target of a transaction and an address parity error occurs, or a data parity error occurs on a write transaction to system memory, it continues the transaction on the PCI bus but aborts internally. The PCI controller does not target-abort in this case.

If the PCI controller is mastering a transaction that terminates with a target-abort, undefined data is returned on a read and write data is lost. An example of a target-abort is shown in Figure 13-48.

An initiator may retry any target disconnect accesses, except target-abort, at a later time starting with the address of the next non-transferred data. Retry is actually a special case of disconnect where no data transfer occurs at all and the initiator must start the entire transaction over again.

13.4.4 Other Bus Operations

The following sections provide information on additional PCI bus operations.

13.4.4.1 Fast Back-to-Back Transactions

In the two types of fast back-to-back transactions, the first type places the burden of avoiding contention on the initiator while the second places the burden on all potential targets. The PCI controller as a target supports both types of fast back-to-back transactions but does not support them as an initiator. The PCI controller as a target has the fast back-to-back enable bit hardwired to one, that is, enabled.

For the first type (governed by the initiator), the initiator may only run a fast back-to-back transaction to the same target. For the second type, when the PCI controller detects a fast-back-to-back operation and did not drive $\overline{\text{PCI\_DEVSEL}}$ in the previous cycle, it delays the assertion of $\overline{\text{PCI\_DEVSEL}}$ and $\overline{\text{PCI\_TRDY}}$ for one cycle to allow the other target to get off the bus.

13.4.4.2 Dual Address Cycles

The PCI controller supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as a target only. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The PCI controller supports single-beat and burst DAC transactions.

13.4.4.3 Data Streaming

The PCI controller provides data streaming for PCI transactions to and from prefetchable memory. In other words, when the PCI controller is a target for a PCI initiated transaction, it supplies or accepts multiple cache lines of data without disconnecting. For PCI transactions to non-prefetchable space, the PCI controller disconnects after the first data phase so streaming cannot occur.

For PCI memory reads, streaming is achieved by performing speculative reads from memory in prefetchable space. A block of memory may be marked as prefetchable by setting the PCI configuration registers bit for the inbound address translation (see Section 13.3.2.14, “PCI Inbound Window Attribute Registers (PIWARn),” for more information) in the following cases:

- When reads do not alter the contents of memory (reads have no side effects)
- When reads return all bytes regardless of the byte enable signals
- When writes can be merged without causing errors

For a memory read command or a memory read line command, the PCI controller reads one cache line from memory. If the transaction crosses a cache line boundary, the PCI controller starts the read of a new cache line. For a memory read multiple command, the PCI controller reads two cache lines from memory. When the PCI transaction finishes the read for the first cache line, the PCI controller performs a speculative read of a third cache line. The PCI controller continues this prefetching until the end of the transaction.

For PCI writes to memory, streaming is achieved by buffering the transaction in the space available within the I/O sequencer. This allows PCI memory writes to execute with no wait states.

A disconnect occurs if the PCI controller runs out of buffer space on writes, or the PCI controller cannot supply consecutive data beats for reads within eight PCI bus clocks of each other. A disconnect also occurs if the transaction crosses a 4-Kbyte page boundary.

13.4.4.4 Host Mode Configuration Access

The PCI controller provides two types of configuration accesses to support hierarchical bridges. To access configuration space, a value is written to the CONFIG_ADDR register specifying which PCI bus, which device, and which configuration register to be accessed.

When the PCI controller sees an access that falls inside the 4 bytes beginning at the CONFIG_DATA address, it checks the enable bit, the device number and the bus number in the CONFIG_ADDR register. If the enable bit is set and the device number is not equal to all ones, a configuration cycle translation is performed. When the device number field is equal to all ones, it has a special meaning (see Section 13.4.4.6, “Special Cycle Command,” for more information).

There are two types of translations supported:

- Type 0 translations—For when the device is on the PCI bus connected to the PCI controller.
- Type 1 translations—For when the device is on another bus somewhere behind the PCI controller.

For type 0 translations, the PCI controller decodes the device number field to assert the appropriate IDSEL line and perform a configuration cycle on the PCI bus with AD[1:0] as 0b00. All 21 IDSEL bits are decoded, starting with bit AD11. That is, if the device number field contains 0b01011, AD11 on the PCI bus is set. The IDSEL lines are bit-wise associated with increasing values for the device number such that AD12 corresponds to 0b01100, and so on up to bit 30 as shown in Table 13-41. AD31 is selected with 0b01010. A device number of 0b11111 indicates a special cycle. Device number 0b00000 is used for configuring the PCI controller itself. Bits 10 through 8 are copied to the PCI bus as an encoded value for components which contain multiple functions. Bits 7 through 2 are also copied onto the PCI bus. The PCI controller implements address stepping on configuration cycles so that the target's PCI_IDSEL, which is connected directly to one of the AD lines, reaches a stable value. This means that a valid address and command are driven on the AD and PCI_C/$\overline{\text{BE}}$ lines one cycle before the assertion of $\overline{\text{PCI\_FRAME}}$.

For type 1 translations, the PCI controller copies the contents of the CONFIG_ADDR register directly onto the PCI address/data lines during the address phase of a configuration cycle, with the exception that AD[1-0] contains 0b01 (not 0b00 as in Type 0 translations).

When the PCI controller is configured as a host device, a local master sometimes needs to perform configuration reads from unpopulated PCI slots (as part of the system configuration). To avoid getting a machine check interrupt, the following steps should be taken:

1. Mask the NORSP bit in the error mask register. See Section 13.3.2.9, "PCI Error Control Register (PCI_ECR)."
2. Perform the PCI configuration reads.
3. Clear the NORSP bit in the error status register.
4. Unmask (write 1) the NORSP bit in the error mask register. See Section 13.3.2.3, "PCI Error Enable Register (PCI_EER)."

13.4.4.5 Agent Mode Configuration Access

When the PCI controller is configured as an agent device, it responds to remote host generated PCI configuration accesses to the PCI interface. This is indicated by decoding the configuration command along with the PCI controller's IDSEL being asserted. A remote host can access the 256-byte PCI configuration area and the memory-mapped configuration registers within the PCI controller.

13.4.4.6 Special Cycle Command

A special cycle command contains no explicit destination address but is broadcast to all PCI agents. Each receiving agent must determine whether the message is applicable to itself. No assertion of $\overline{\text{PCI\_DEVSEL}}$ in response to a special cycle command is necessary.

A special cycle command is like any other bus command in that it has an address phase and a data phase. The address phase starts like all other commands with the assertion of $\overline{\text{PCI\_FRAME}}$ and completes when

$\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated. Special cycles terminate with a master-abort. (In the special cycle case, the received-master-abort bit in the configuration status register is not set.)

The address phase contains no valid information other than the command field. Even though there is no explicit address, the address/data lines are driven to a stable state and parity is generated. During the data phase, the address/data lines contain the message type and an optional data field. The message is encoded on the sixteen least-significant bits (AD[15:0]). The data field is encoded on AD[31:16]. When running a special cycle, the message and data are valid on the first clock $\overline{\text{PCI\_IRDY}}$ is asserted.

When the PCI_CONFIG_ADDRESS register is written with a value so that the bus number matches the bridge bus, the device number is all ones, the function number is all ones, and the register number is zero. The next time the PCI_CONFIG_DATA register is accessed, the PCI controller executes either a special cycle or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is written, the PCI controller generates a special cycle encoding on the command/byte enable lines during the address phase and drives the data from the PCI_CONFIG_DATA register onto the address/data lines during the first data phase.

If the bus number field of the PCI_CONFIG_ADDRESS does not match one of the PCI controller bus numbers, the PCI controller passes the write to PCI_CONFIG_DATA through to the PCI bus as a type 1 configuration cycle as it does any other time the bus number field does not match.

Table 13-44. Special Cycle Commands

Address (AD[15–0])	Message Type	Description
0x0000	SHUTDOWN (SLEEP)	Indicates the processor is entering its most power saving mode
0x0001	HALT (DOZE)	Indicates the processor is entering a power save mode where address decoding is still available
0x0002–0xFFFF	—	Reserved for future commands

13.4.4.7 Interrupt Acknowledge

When the PCI_CONFIG_ADDRESS register is written with a value such that the bus number is 0x00, the device number is all ones, the function number is all ones, and the register number is zero, the next time the PCI_CONFIG_DATA register is accessed the PCI controller does either a special cycle command or an interrupt acknowledge command. When the PCI_CONFIG_DATA register is read, the PCI controller generates an interrupt acknowledge command encoding on the command/byte enable lines during the address phase. During the address phase, AD[31:0] do not contain a valid address but are driven with stable data and valid parity (PCI_PAR). During the data phase, the byte enable signals determine which bytes are involved in the transaction. The interrupt vector must be returned when $\overline{\text{PCI\_TRDY}}$ is asserted.

An interrupt acknowledge transaction can also be issued on the PCI bus by reading from the PCI_INT_ACK register.

13.4.5 Error Functions

This section describes PCI bus errors.

13.4.5.1 Parity

During valid 32-bit address and data transfers, parity covers all 32 address/data lines and the 4 command/byte enable lines regardless of whether or not all lines carry meaningful information. Byte lanes not actually transferring data are driven with stable (albeit meaningless) data and are included in the parity calculation. During configuration, special cycle or interrupt acknowledge commands, some address lines are not defined but are still driven to stable values and included in the parity calculation.

Even parity is calculated for all PCI operations: the value of PCI_PAR is generated such that the number of ones on PCI_AD[31:0], PCI_C/$\overline{\text{BE}}$[3:0] and PCI_PAR equals an even number. The PCI_PAR signal is driven when the address/data lines are driven and follow the corresponding address or data by one clock.

The PCI controller checks the parity after all valid address phases (the assertion of $\overline{\text{PCI\_FRAME}}$) and for valid data transfers ($\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ asserted) involving the PCI controller. When an address or data parity error is detected, the detected-parity-error bit in the configuration space status register is set (see Section 13.3.3.4, “PCI Status Configuration Register.”)

13.4.5.2 Error Reporting

Except for setting the detected-parity-error bit, all parity error reporting and response is controlled by the parity-error-response bit (see Section 13.3.3.3, “PCI Command Configuration Register,” for more information). If the parity-error-response bit is cleared, the PCI controller completes all transactions regardless of parity errors (address or data). If the bit is set, the PCI controller asserts $\overline{\text{PCI\_PERR}}$ two clocks after the actual data transfer in which a data parity error is detected, and keeps $\overline{\text{PCI\_PERR}}$ asserted for one clock. When acting as an initiator during a read transaction or as a target involved in a write to system memory the PCI controller asserts $\overline{\text{PCI\_PERR}}$.

Figure 13-51 shows the possible assertion points for PCI_PERR if the PCI controller detects a data parity error.

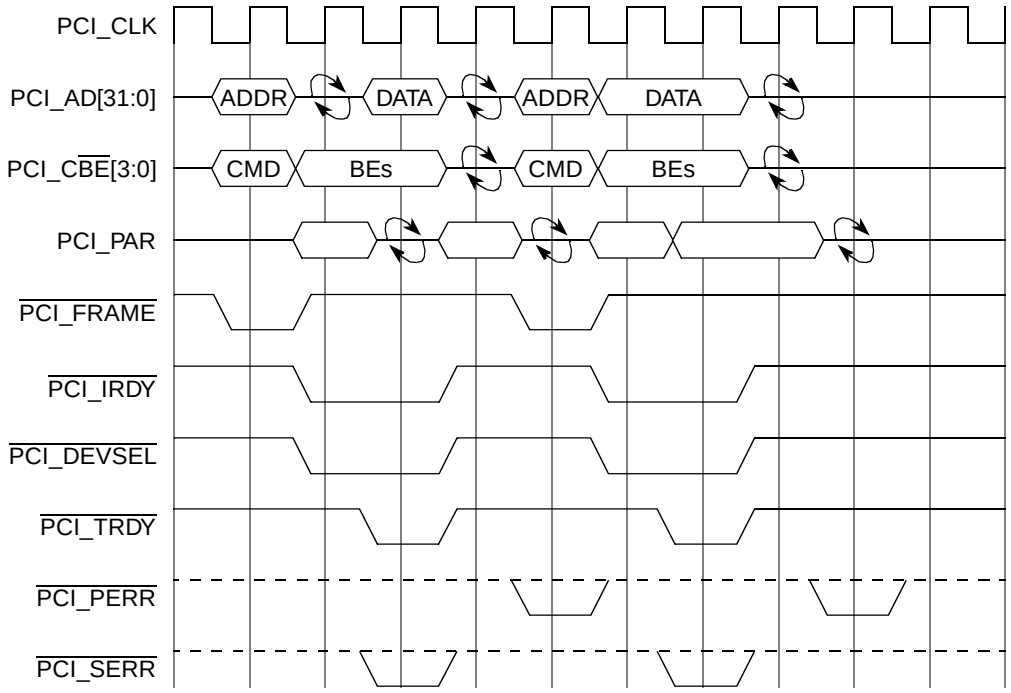


Figure 13-51. PCI Parity Operation

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.

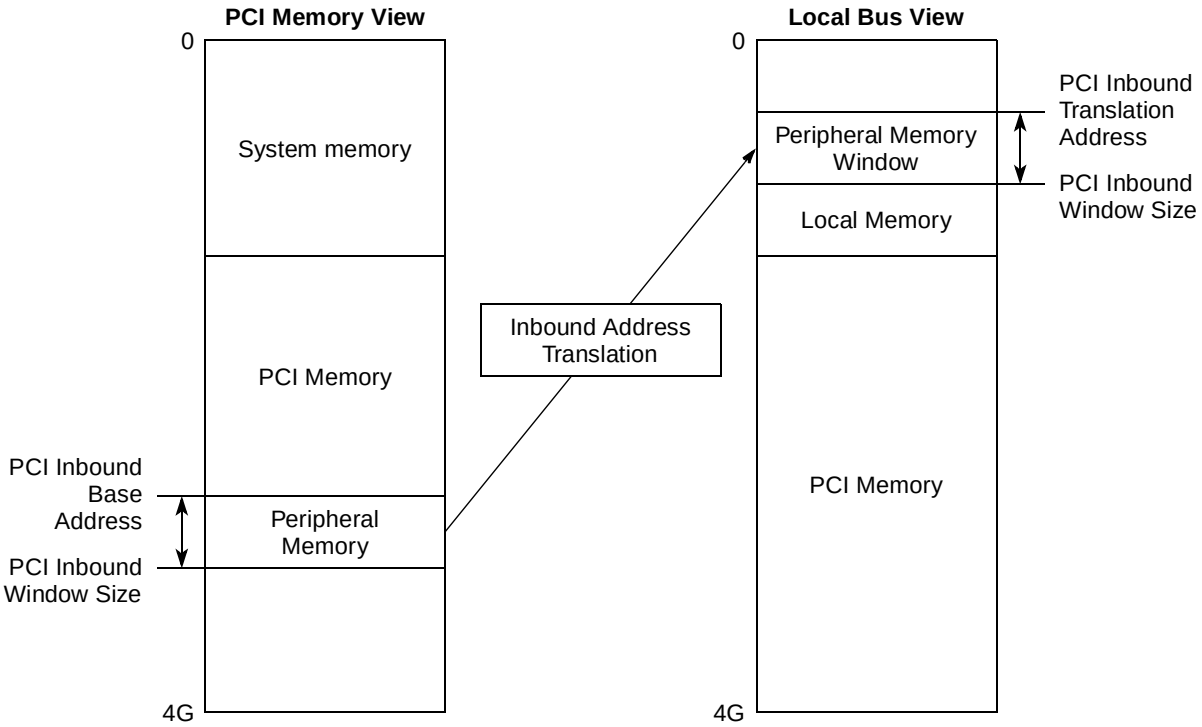


Figure 13-52. Inbound PCI Memory Address Translation

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) 2.4

This chapter describes the functionality of the device's integrated security engine (SEC 2.4). It addresses the following topics:

14.1 Overview

The SEC 2.4 is designed to offload computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the device processor core. It is optimized to process all the algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i. The SEC 2.4 is derived from integrated security cores found in other members of the PowerQUICC family, including SEC 1.0, the version implemented in the MPC8272/MPC8248.

The security engine's execution units (EUs) and primary features include the following:

- PKEU—Public key execution unit that supports the following:
  - RSA and Diffie-Hellman algorithms
    - Programmable field size up to 2048 bits
  - Elliptic curve cryptography
    - F_2m and F(p) modes
    - Programmable field size up to 511 bits
- DEU—Data encryption standard execution unit
  - DES, 3DES algorithms
  - Two key (K1, K2) or three key (K1, K2, K3) for 3DES
  - ECB and CBC modes for both DES and 3DES
- AESU—Advanced encryption standard unit
  - Implements the Rijndael symmetric-key cipher

  - ECB, CBC, CCM, and counter modes
  - 128-, 192-, 256-bit key lengths
- AFEU—ARC four execution unit
  - Implements a stream cipher compatible with the RC4 algorithm
  - 40- to 128-bit programmable key
- MDEU—Message digest execution unit
  - SHA with 160-, 224-, or 256-bit message digest
  - MD5 with 128-bit message digest
  - HMAC with either algorithm
- RNG—Random number generator
- XOR parity generation accelerator for RAID applications
- Master/slave logic, with DMA capability
  - 32-bit address/64-bit data
  - Master interface allows multiple pipelined requests
  - DMA blocks can be on any byte boundary
- Four channels, each supporting a queue of commands (descriptor pointers)
  - Dynamic assignment of crypto-execution units through an integrated controller
  - 256-byte buffer FIFOs on data input and output paths of each execution unit, with flow control for large data sizes
- Scatter/Gather capability
  - Gather capability enables the SEC 2.4 to concatenate multiple segments of memory when reading input data
  - Similarly, scatter capability enables SEC 2.4 to write to multiple segments of memory when writing output data

14.2 Architecture Overview

The SEC 2.4 (henceforth referred to as SEC) can act as a master on the internal bus. This allows the SEC to alleviate the data movement bottleneck normally associated with slave-only cores. The host processor accesses the SEC through its device drivers, using system memory for data storage. The SEC resides in the peripheral memory map of the processor; therefore, when an application requires cryptographic functions, it simply creates descriptors for the SEC that define the cryptographic function to be performed and the location of the data. The SEC's bus-mastering capability permits the host processor to set up a crypto-channel with a few short register writes, leaving the SEC to perform reads and writes on system memory to complete the required task.

Figure 14-1 shows that the SEC communicates with other modules via the internal bus.

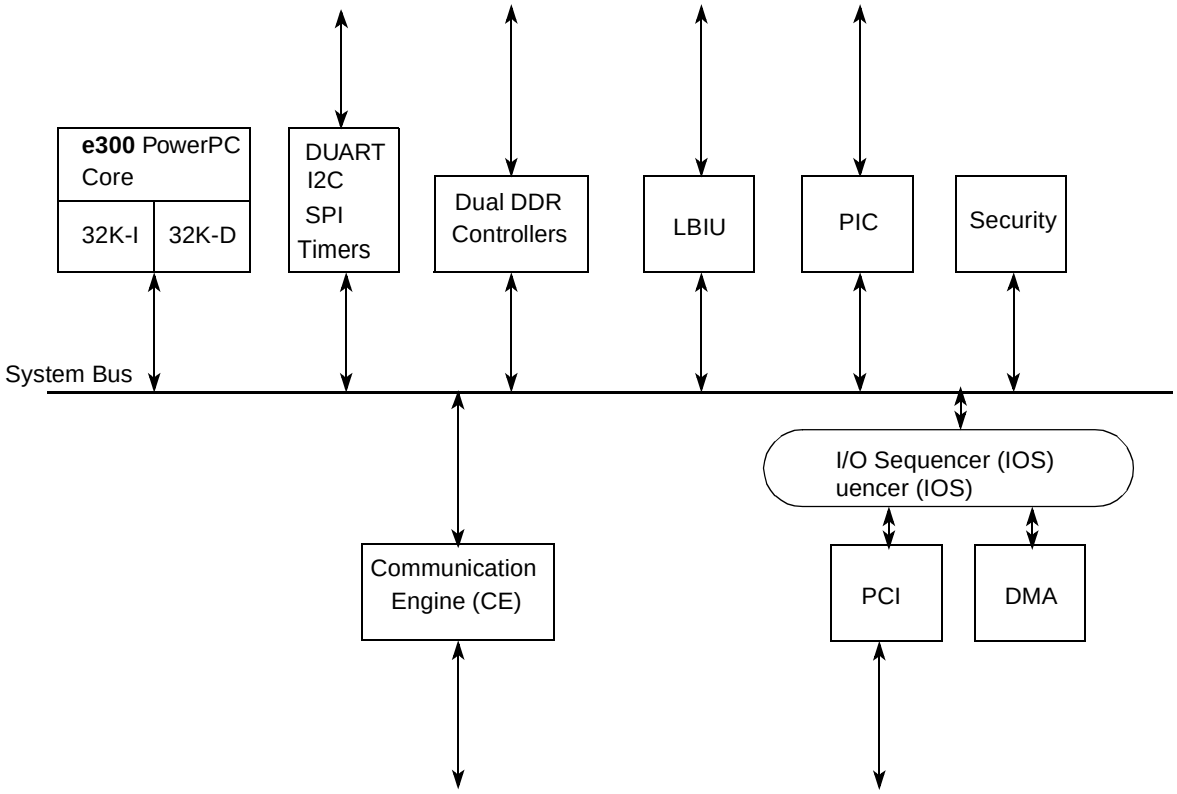


Figure 14-1. SEC Connected to MPC8360E Internal Bus

A block diagram of the SEC internal architecture is shown in Figure 14-2. The bus interface module is designed to transfer 64-bit words between the bus and any register inside the SEC.

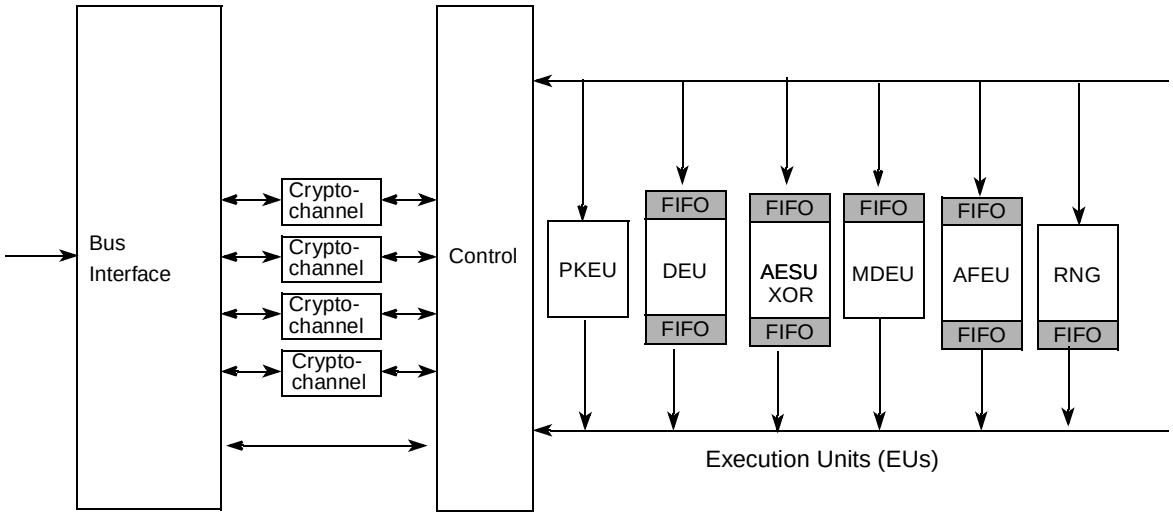


Figure 14-2. SEC Functional Modules

An operation begins when the host writes a descriptor pointer to the fetch FIFO in one of the four SEC channels. From this point on, the channel directs the sequence of operations. The channel uses the descriptor pointer to read the descriptor, then decodes the first word of the descriptor to determine the operation to be performed to select and the crypto-execution units needed to perform it. The channel requests the controller to assign the needed crypto-execution units. Next the channel requests that the controller fetch the keys, and data from locations specified in the rest of the descriptor. The controller satisfies the requests by making requests to the master interface following the programmable priority scheme. Data is fed into the execution units through their registers and input FIFOs. The execution units read from their input FIFOs and write processed data to their output FIFOs. The channel requests the controller to write data from the output FIFOs and registers back to system memory through the master/slave interface.

For most packets, the entire payload is too long to fit in an execution unit's input or output FIFO. The SEC then uses a flow control scheme for reading and writing data. The channel directs the controller to read bursts of input as necessary to keep refilling the input FIFO, until the entire payload has been fetched. Similarly, the channel directs the controller to write bursts of output whenever enough accumulates in the execution unit's output FIFO.

14.2.1 Data Packet Descriptors

As a crypto-acceleration block, the SEC controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through descriptors. A descriptor specifies a cryptographic function to be performed and contains pointers to all necessary input data and to the places where output data is to be written. Some descriptor types perform multiple functions to facilitate particular protocols. A data packet descriptor is diagrammed in Table 14-1.

Table 14-1. Example Data Packet Descriptor

Field Name	Value/Type	Description
DPD_DES_CTX_CRYPT	TBD	Representative header for DES using context to encrypt
LEN_CTXIN PTR_CTXIN	Length Pointer	Number of bytes to be written Pointer to context (IV) to be written into DES engine
LEN_KEY PTR_KEY	Length Pointer	Number of bytes in key Pointer to block cipher key
LEN_DATAIN PTR_DATAIN	Length Pointer	Number of bytes of data to be ciphered Pointer to data to perform cipher upon
LEN_DATAOUT PTR_DATAOUT	Length Pointer	Number of bytes of data after ciphering Pointer to location where cipher output is to be written
LEN_CTXOUT PTR_CTXOUT	Length Pointer	Length of output context (IV) Pointer to location where altered context is to be written
Null length Null pointer	Length Pointer	Zeros for fixed length descriptor filter Zeros for fixed length descriptor filter
Null length Null pointer	Length Pointer	Zeros for fixed length descriptor filter Zeros for fixed length descriptor filter

Each descriptor contains eight long-words (64 bits each), consisting of the following:

- One long-word of header—The header describes the required services and encodes information indicating which EUs to use and which modes to set. It also indicates if notification should be sent to the host when the descriptor operation is complete.
- Seven long-words containing pointers and lengths used to locate input or output data. Each pointer can either point directly to the data, or can point to a link table that lists a set of data segments to be concatenated.

For more information, refer to Section 14.4, "Descriptor Overview."

14.2.2 Execution Units (EUs)

Execution unit (EU) is the generic term for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i processing, and can work together to perform high-level cryptographic tasks. The SEC's execution units are as follows:

- PKEU—For computing asymmetric-key operations, including modular exponentiation (and other modular arithmetic functions) or ECC point arithmetic
- DEU—For performing block cipher, symmetric-key cryptography using DES and 3DES
- AFEU—For performing RC-compatible stream cipher symmetric-key cryptography
- AESU—For performing the advanced encryption standard algorithm and XOR acceleration
- MDEU—For performing security hashing using MD-5, SHA-1, or SHA-256
- RNG—For random number generation

Each EU is described in detail in Section 14.5, "Execution Units."

14.2.2.1 Public Key Execution Unit (PKEU)

The PKEU can perform many advanced mathematical functions to support both RSA and ECC public-key cryptographic algorithms. ECC is supported in both F_2m (polynomial-basis) and F(p) modes. This EU supports all levels of functions to assist the host microprocessor to perform its desired cryptographic task. For example, at the highest level, the accelerator performs modular exponentiations to support RSA and performs point multiplies to support ECC. At the lower levels, the PKEU can perform simple operations such as modular multiplies. For more information, refer to Section 14.5.1, "Public Key Execution Unit (PKEU)."

14.2.2.1.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 160 and 512 bits in programmable increments of 8, with each value i supporting all actual field sizes from 8i – 7 to 8i. The result is hardware supporting a wide range of cryptographic security. Larger field/modulus sizes result in greater security but lower performance; processing time is determined by field or modulus size. For example, a field size of 160 is roughly equivalent to the security provided by 1024 bit RSA. A field size set to 208 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations, which are addition, multiplication, inverse, and squaring. Point add and double each use all four finite field operations. Similarly, point multiplication uses all EC point operations as well as the finite field operations. All these functions are supported both in modular arithmetic and polynomial basis finite fields.

14.2.2.1.2 Modular Exponentiation Operations

The PKEU can also perform ordinary integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the SEC's PKEU include the following:

- R^2 mod N
- A'^E mod N
- (A × B) R^{-1} mod N
- (A × B) R^{-2} mod N
- (A + B) mod N
- (A – B) mod N

Where the following variable definitions: A′ = AR mod N, N is the modulus vector, A and B are input vectors, E is the exponent vector, R is 2^s, where s is the bit length of the N vector rounded up to the nearest multiple of 32.

The PKEU can perform modular arithmetic on operands up to 2048 bits in length. The modulus must be larger than or equal to 97 bits (13 bytes). This is not seen as a limitation since for sizes smaller than this, no useful cryptographic application exists. Furthermore, for small data sizes the overhead of using a hardware accelerator would not be justified. The PKEU uses the Montgomery modular multiplication algorithm to perform core functions. The addition and subtraction functions exist to help support known methods of the Chinese remainder theorem (CRT) for efficient implementation of the RSA algorithm.

14.2.2.2 Data Encryption Standard Execution Unit (DEU)

The DES execution unit (DEU) performs bulk data encryption/decryption, in compliance with the data encryption standard algorithm (ANSI x3.92). The DEU can also compute 3DES, an extension of the DES algorithm in which each 64-bit input block is processed three times. The SEC supports 2-key (K1=K3) or 3-key 3DES.

The DEU operates by permuting 64-bit data blocks with a shared 56-bit key and an initialization vector (IV). The SEC supports two modes of operation: electronic code book (ECB) and cipher clock chaining (CBC).

For more information, refer to Section 14.5.2, "Data Encryption Standard Execution Unit (DEU)."

14.2.2.3 ARC Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning a byte of plain text is encrypted with a key to produce a byte of ciphertext. The key is variable length and the AFEU supports key lengths from 8 to 128 bits (in byte increments), providing a wide range of security strengths. ARC4 is a symmetric algorithm, meaning each of the two communicating parties share the same key.

For more information, refer to Section 14.5.3, "ARC Four Execution Unit (AFEU)."

14.2.2.4 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the Advanced Encryption Standard (Rijndael) algorithm. The AESU executes on 128-bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AESA is a symmetric-key algorithm; the sender and receiver use the same key for both encryption and decryption. The session key and IV are supplied to the AESU module prior to encryption. The processor supplies data to the module that is processed as a 128-bit input. The AESU operates in ECB, CBC, CTR, and CCM modes.

The AESU is also used for performing the eXclusive OR (XOR) operation used to generate parity data for RAID storage applications. When operating in this mode, no session keys are involved, and the AESU XORs up to 3 data streams at a time to produce parity data.

For more information, refer to Section 14.5.6, "Advanced Encryption Standard Execution Units (AESU)."

14.2.2.5 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using either the MD5, SHA-1 or SHA-256 algorithms for bulk data hashing. With any hash algorithm, the larger message is mapped onto a smaller output space; therefore, collisions are possible, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash; the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
- SHA-224 and SHA-256 are cryptographic hash functions that provide integrity protection against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

For more information, refer to Section 14.5.4, "Message Digest Execution Unit (MDEU)."

14.2.2.6 Random Number Generator (RNG)

The RNG is a functional block capable of generating 64-bit random numbers. It is designed to comply with FIPS 140-1 standards for randomness and non-determinism.

Because many cryptographic algorithms use random numbers as a source for generating a secret value (a nonce), it is desirable to have a private RNG for use by the SEC. The anonymity of each random number must be maintained, as well as the unpredictability of the next random number. The FIPS-140 ‘common criteria’ compliant private RNG allows the system to develop random challenges or random secret keys. The secret key can thus remain hidden from even the high-level application code, providing an added measure of physical security.

For more information, refer to Section 14.5.5, “Random Number Generator (RNG).”

14.2.3 Crypto-Channels

The SEC includes four crypto-channels that manage data and EU function. Each channel consists of the following:

- A fetch FIFO, which holds a queue of pointers to descriptors waiting to be serviced
- A configuration register, which allows the user a number of options for SEC event signaling.
- Control registers containing information about the transaction in process
- A status register containing an indication of the last unfulfilled bus request
- A descriptor buffer memory used to store the active descriptor
- Scatter and gather link table buffer memory used to store the active link table

Whenever a channel is idle and its fetch FIFO is non-empty, the channel reads the next descriptor pointer from the fetch FIFO. Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. To service this descriptor, the channel directs execution of the following steps.

1. Analyze the descriptor header to determine the cryptographic services required and request use of the appropriate EU(s) from the controller.
2. Wait for the controller to grant access to the required EU(s).
3. Set the appropriate mode bits in the EU(s) for the required service.
4. Fetch ‘data parcels’ using pointers from the descriptor buffer, and place them in either an EU input FIFO or EU registers (as appropriate). The term data parcel refers here to any input or output of a cryptographic process, such as a key, hash result, input context, output context, or text-data. ‘Context’ refers to either an IV or other internal EU state that can be read out or loaded in. ‘Text-data’ refers to plaintext or ciphertext to be operated on.
5. If the data size is greater than EU FIFO size, continue fetching input data, and writing output data to memory.
6. Wait for EU(s) to complete processing.
7. Upon completion, unload results from output FIFOs and context registers and write them to external memory using pointers in the descriptor buffer.
8. If multiple services are requested, go back to step 3.
9. Release the EUs. (Note that in previous Freescale security co-processors, it was possible to reserve an EU for use on multiple descriptors. With the added capabilities in SEC 2.4, such ‘static’ assignments are no longer necessary and are not supported. EUs are always released at the completion of a descriptor.)
10. If ‘done notification’ is enabled in the descriptor header, perform this notification.

The channel can signal to the host that it is done with a descriptor via interrupt or by a writeback of the descriptor header into host memory. In the case of writeback, the value written back is identical to the header that was read, with the exception that a done field is set to all 1s. In addition, the channel can be configured to write back other status fields that indicate the result of ICV checking (if any). The user can opt to do this signaling at end of every descriptor, or at end of selected descriptors. For more about configuring signaling, see Section 14.6.1.1, "Crypto-Channel Configuration Register (CCCR)."

Many security protocols involve both encryption and hashing of packet payloads. To accomplish this without requiring two passes through the data, channels can configure data flows through more than one EU. In such cases, one EU is designated the primary EU, and the other as the secondary EU. The primary EU receives its data from memory through the controller, and the secondary EU receives its data by 'snooping' the SEC buses.

There are two types of snooping.

- Input data can be fed to the primary EU and the same input data snooped by the secondary EU. This is called 'in-snooping.'
- Output data from the primary EU can be snooped by the secondary EU. This is called 'out-snooping.'

In the SEC, the secondary EU is always the MDEU.

For more information, refer to Section 14.6, "Crypto-Channels."

14.2.4 SEC Controller

The SEC controller manages on-chip resources, including the individual execution units (EUs), FIFOs, the master/slave interface, and the internal buses that connect all the various modules. The controller receives service requests from the master/slave interface and various crypto-channels, and schedules the required activities. The controller can configure each of the on-chip resources in two modes:

- Host-controlled access—The host is directly responsible for all data movement into and out of the resource. This mode is typically only used in debug.
- Channel-controlled access—A channel can request a particular service from any available execution unit. This is the normal operating condition.

The system bus interface and access to system memory are critical factors in performance, and the 64-bit master/slave interface of the SEC controller allows it to achieve performance unattainable on secondary buses.

14.2.4.1 Host-Controlled Access

All execution units (EU) can be used entirely through register read/write access. The SEC operates as a slave, and the host must target write the information typically provided through the descriptor into the appropriate registers and FIFOs of the SEC. This mode is more CPU intensive, and requires a great deal of familiarity with the SEC registers. It is recommended that host-controlled access be used only for operations using a single EU, and for debug purposes.

For more information, refer to Section 14.7, "Controller."

14.2.4.2 Channel-Controlled Access

Processing begins when a descriptor pointer is written to the fetch FIFO of one of the channels. Based on the services requested by the descriptor header, the channel asks the controller to assign the necessary EUs to that channel. If all appropriate EUs are already reserved by other channels, the channel stalls and waits to fetch data until an appropriate EU is available. If multiple channels simultaneously request the same EU, the EU is assigned on a weighted priority or round-robin basis.

Once the required EU has been reserved, the channel requests that the controller fetch and load the appropriate data. The controller acts as a master on the system bus, reading and writing on byte boundaries. The channel operates the EU, and makes further requests to the controller to write output data to system memory. When the descriptor processing is complete, the channel asks the controller to release the EU for use by other channels.

14.2.5 Bus Interface

The master/slave interface manages communication between the SEC's internal execution units and the device internal bus. All on-chip resources are memory mapped, and the slave accesses to the SEC may be addressed on byte boundaries. The SEC performs initiator reads on byte boundaries and internally adjusts the data to place on word boundaries as appropriate. Access to system memory is a critical factor in performance, and the 64-bit master/slave interface of the SEC allows it to achieve performance unattainable on secondary buses.

For more information, refer to Section 14.8, "Bus Interface."

14.3 Configuration of Internal Memory Space

Table 14-2 shows the base address map, while Table 14-3 provides the address map, including all registers in the execution units. The 18-bit SEC address bus value is shown. These address values are offsets from the internal memory mapped registers base address, as programmed in the IMMRBAR. See Section 2.3, "Complete IMMR Map," for more information.

Note that these tables show modulo-8 addresses; the three least significant address bits that are used to select bytes within 64-bit-words are not shown.

Table 14-2. SEC Base Address Map

Offset	Module	Description	Type	Section/Page
0x3_0000–0x3_0FFF	—	Reserved	—	—
0x3_1000–0x3_10FF	Controller	Arbiter/controller control register space	Resource control	14.7/14-93
0x3_1100–0x3_11FF	Channel_1	Crypto-channel 1	Data control	14.6/14-81
0x3_1200–0x3_12FF	Channel_2	Crypto-channel 2		
0x3_1300–0x3_13FF	Channel_3	Crypto-channel 3		
0x3_1400–0x3_14FF	Channel_4	Crypto-channel 4		

Table 14-2. SEC Base Address Map (continued)

Offset	Module	Description	Type	Section/Page
0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	14.5.2/14-34
0x3_4000–0x3_4FFF	AESU	AES execution unit		14.5.6/14-68
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		14.5.4/14-51
0x3_8000–0x3_8FFF	AFEU	ARC four execution unit		14.5.3/14-42
0x3_A000–0x3_AFFF	RNG	Random number generator		14.5.5/14-63
0x3_C000–0x3_CFFF	PKEU	Public key execution unit		14.5.1/14-26

Table 14-3 shows the system address map showing all functional registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 14-3. SEC Address Map

Offset	Register	Access	Reset	Section/Page
Controller Registers				
0x3_1008	IMR—Interrupt mask register	R/W	0x0000_0000_0000_0000	14.7.2.1/14-94
0x3_1010	ISR—Interrupt status register	R	0x0000_0000_0000_0000	14.7.2.2/14-96
0x3_1018	ICR—Interrupt clear register	W	0x0000_0000_0000_0000	14.7.2.3/14-96
0x3_1020	ID—Identification register	R	0x0000_0000_0000_0040	14.7.2.4/14-98
0x3_1028	EUASR—EU assignment status register	R	0xF0F0_F0F0_00FF_F0F0	14.7.2/14-93
0x3_1030	MCR—Master control register	R/W	0000_0000_0000_0000	14.7.2.6/14-99
Channel 1				
0x3_1108	CCCR1—Crypto-channel 1 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1110	CCPSR1—Crypto-channel 1 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1140	CDPR1—Crypto-channel 1 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1148	FF1—Crypto-channel 1 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1180–0x3_11BF	DBn—Crypto-channel 1descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Channel 2				
0x3_1208	CCCR2—Crypto-channel 2 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1210	CCPSR2—Crypto-channel 2 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1240	CDPR2—Crypto-channel 2 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1248	FF2—Crypto-channel 2 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1280–0x3_12BF	DBn—Crypto-channel 2 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
Channel 3				
0x3_1308	CCCR3—Crypto-channel 3 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1310	CCPSR3—Crypto-channel 3 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1340	CDPR3—Crypto-channel 3 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1348	FF3—Crypto-channel 3 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1380–0x3_13BF	DBn—Crypto-channel 3 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Channel 4				
0x3_1408	CCCR4—Crypto-channel 4 configuration register	R/W	0x0000_0000_0000_0000	14.6.1.1/14-82
0x3_1410	CCPSR4—Crypto-channel 4 pointer status register	R	0x0000_0000_0000_0007	14.6.1.2/14-85
0x3_1440	CDPR4—Crypto-channel 4 current descriptor pointer register	R	0x0000_0000_0000_0000	14.6.1.3/14-90
0x3_1448	FF4—Crypto-channel 4 fetch FIFO address register	W	0x0000_0000_0000_0000	14.6.1.4/14-90
0x3_1480–0x3_14BF	DBn—Crypto-channel 4 descriptor buffers[0–7]	R	0x0000_0000_0000_0000	14.6.1.5/14-91
Data Encryption Standard Execution Unit (DEU)				
0x3_2000	DEUMR—DEU mode register	R/W	0x0000_0000_0000_0000	14.5.2.1/14-35
0x3_2008	DEUKSR—DEU key size register	R/W	0x0000_0000_0000_0000	14.5.2.2/14-36
0x3_2010	DEUDSR—DEU data size register	R/W	0x0000_0000_0000_0000	14.5.2.3/14-36
0x3_2018	DEURCR—DEU reset control register	R/W	0x0000_0000_0000_0000	14.5.2.4/14-37
0x3_2028	DEUSR—DEU status register	R	0x0000_0000_0000_0000	14.5.2.5/14-37
0x3_2030	DEUISR—DEU interrupt status register	R	0x0000_0000_0000_0000	14.5.2.6/14-38
0x3_2038	DEUICR—DEU interrupt control register	R/W	0x0000_0000_0000_3000	14.5.2.7/14-40
0x3_2050	DEUEUG—DEU EU-Go register	W	0x0000_0000_0000_0000	14.5.2.8/14-41
0x3_2100	DEUIV—DEU initialization vector register	R/W	0x0000_0000_0000_0000	14.5.2.9/14-42
0x3_2400	DEUK1—DEU key register 1	W	—	14.5.2.10/14-42
0x3_2408	DEUK2—DEU key register 2	W	—	14.5.2.10/14-42
0x3_2410	DEUK3—DEU key register 3	W	—	14.5.2.10/14-42
0x3_2800–0x3_2FFF	DEU FIFO	R/W	0x0000_0000_0000_0000	14.5.2.11/14-42
Advanced Encryption Standard Execution Unit (AESU)				
0x3_4000	AESUMR—AESU mode register	R/W	0x0000_0000_0000_0000	14.5.6.1/1414-68
0x3_4008	AESUKSR—AESU key size register	R/W	0x0000_0000_0000_0000	14.5.6.2/1414-71

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_4010	AESUDSR—AESU data size register	R/W	0x0000_0000_0000_0000	14.5.6.3/1414-71
0x3_4018	AESURCR—AESU reset control register	R/W	0x0000_0000_0000_0000	14.5.6.4/14-72
0x3_4028	AESUSR—AESU status register	R	0x0000_0000_0000_0000	14.5.6.5/14-73
0x3_4030	AESUISR—AESU interrupt status register	R	0x0000_0000_0000_0000	14.5.6.6/14-74
0x3_4038	AESUICR—AESU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.6.7/14-75
0x3_4050	AESUEMR—AESU end of message register	W	0x0000_0000_0000_0000	14.5.6.8/14-76
0x3_4100	AESU context memory registers	R/W	0x0000_0000_0000_0000	14.5.6.9/14-77
0x3_4400–0x3_4408	AESU key memory	R/W	0x0000_0000_0000_0000	14.5.6.9.5/14-81
0x3_4800–0x3_4FFF	AESU FIFO	R/W	0x0000_0000_0000_0000	14.5.6.9.6/14-81
Message Digest Execution Unit (MDEU)				
0x3_6000	MDEUMR—MDEU mode register	R/W	0x0000_0000_0000_0000	14.5.4.1/14-51
0x3_6008	MDEUKSR—MDEU key size register	R/W	0x0000_0000_0000_0000	14.5.4.3/14-55
0x3_6010	MDEUDSR—MDEU data size register	R/W	0x0000_0000_0000_0000	14.5.4.4/14-56
0x3_6018	MDEURCR—MDEU reset control register	R/W	0x0000_0000_0000_0000	14.5.4.5/14-56
0x3_6028	MDEUSR—MDEU status register	R	0x0000_0000_0000_0000	14.5.4.6/14-57
0x3_6030	MDEUISR—MDEU interrupt status register	R	0x0000_0000_0000_0000	14.5.4.7/14-58
0x3_6038	MDEUICR—MDEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.4.8/14-59
0x3_6040	MDEUICVSR—MDEU ICV size register	R/W	0x0000_0000_0000_0000	14.5.4.9/14-60
0x3_6050	MDEUEUG—MDEU EU-Go register	W	0x0000_0000_0000_0000	14.5.4.10/14-61
0x3_6100–0x3_6120	MDEU context memory registers	R/W	0x0000_0000_0000_0000	14.5.4.11/14-61
0x3_6400–0x3_647F	MDEU key memory	W	0x0000_0000_0000_0000	14.5.4.12/14-62
0x3_6800–0x3_6FFF	MDEU FIFO	W	0x0000_0000_0000_0000	14.5.4.13/14-63
ARC Four Execution Unit (AFEU)				
0x3_8000	AFEUMR—AFEU mode register	R/W	0x0000_0000_0000_0000	14.5.3.1/14-43
0x3_808	AFEUKSR—AFEU key size register	R/W	0x0000_0000_0000_0000	14.5.3.3/14-44
0x3_8010	AFEUDSR—AFEU data size register	R/W	0x0000_0000_0000_0000	14.5.3.4/14-45
0x3_8018	AFEURCR—AFEU reset control register	R/W	0x0000_0000_0000_0000	14.5.3.5/14-46
0x3_8028	AFEUSR—AFEU status register	R	0x0000_0000_0000_0000	14.5.3.6/14-46
0x3_8030	AFEUISR—AFEU interrupt status register	R	0x0000_0000_0000_0000	14.5.3.7/14-47

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_8038	AFEUICR—AFEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.3.8/14-49
0x3_8050	AFEUEMR—AFEU end of message register	W	0x0000_0000_0000_0000	14.5.3.9/14-50
0x3_8100–0x3_81FF	AFEU context memory registers	R/W	0x0000_0000_0000_0000	14.5.3.10.1/14-50
0x3_8200	AFEU context memory pointers	R/W	0x0000_0000_0000_0000	14.5.3.10.2/14-51
0x3_8400	AFEUK0—AFEU key register 0	W	—	14.5.3.11/14-51
0x3_848	AFEUK1—AFEU key register 1	W	—	14.5.3.11/14-51
0x3_8800–0x3_8FFF	AFEU FIFO	R/W	0x0000_0000_0000_0000	14.5.3.11.1/14-51
Random Number Generator (RNG)				
0x3_A000	RNGMR—RNG mode register	R/W	0x0000_0000_0000_0000	14.5.5.1/14-63
0x3_A010	RNGDSR—RNG data size register	R/W	0x0000_0000_0000_0000	14.5.5.2/14-64
0x3_A018	RNGRCR—RNG reset control register	R/W	0x0000_0000_0000_0000	14.5.5.3/14-65
0x3_A028	RNGSR—RNG status register	R	0x0000_0000_0000_0000	14.5.5.4/14-65
0x3_A030	RNGISR—RNG interrupt status register	R	0x0000_0000_0000_0000	14.5.5.5/14-66
0x3_A038	RNGICR—RNG interrupt control register	R/W	0x0000_0000_0000_1000	14.5.5.6/14-67
0x3_A050	RNGEUG—RNG EU-Go register	W	0x0000_0000_0000_0000	14.5.5.7/14-68
0x3_A800–0x3_AFFF	RNG FIFO	R	0x0000_0000_0000_0000	14.5.5.8/14-68
Public Key Execution Unit (PKEU)				
0x3_C000	PKEUMR—PKEU mode register	R/W	0x0000_0000_0000_0000	14.5.1.1/14-26
0x3_C008	PKEUKSR—PKEU key size register	R/W	0x0000_0000_0000_0000	14.5.1.2/14-28
0x3_C010	PKEUDSR—PKEU data size register	R/W	0x0000_0000_0000_0000	14.5.1.3/14-28
0x3_C018	PKEURCR—PKEU reset control register	R/W	0x0000_0000_0000_0000	14.5.1.5/14-29
0x3_C028	PKEUSR—PKEU status register	R	0x0000_0000_0000_0000	14.5.1.6/14-30
0x3_C030	PKEUISR—PKEU interrupt status register	R	0x0000_0000_0000_0000	14.5.1.7/14-31
0x3_C038	PKEUICR—PKEU interrupt control register	R/W	0x0000_0000_0000_1000	14.5.1.8/14-32
0x3_C040	PKEUABS—PKEU AB size register	R/W	0x0000_0000_0000_0000	14.5.1.3/14-28
0x3_C050	PKEUEUG—PKEU EU-Go	W	0x0000_0000_0000_0000	14.5.1.9/14-33

Table 14-3. SEC Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C200–0x3_C23F	PKEU parameter memory A0	R/W	0x0000_0000_0000_0000	14.5.1.10/14-34
0x3_C240–0x3_C27F	PKEU parameter memory A1	R/W	0x0000_0000_0000_0000	
0x3_C280–0x3_C2BF	PKEU parameter memory A2	R/W	0x0000_0000_0000_0000	
0x3_C2C0–0x3_C2FF	PKEU parameter memory A3	R/W	0x0000_0000_0000_0000	
0x3_C300–0x3_C33F	PKEU parameter memory B0	R/W	0x0000_0000_0000_0000	
0x3_C340–0x3_C37F	PKEU parameter memory B1	R/W	0x0000_0000_0000_0000	
0x3_C380–0x3_C3BF	PKEU parameter memory B2	R/W	0x0000_0000_0000_0000	
0x3_C3C0–0x3_C3FF	PKEU parameter memory B3	R/W	0x0000_0000_0000_0000	
0x3_C400–0x3_C4FF	PKEU parameter memory E	W	0x0000_0000_0000_0000	
0x3_C800–0x3_C8FF	PKEU parameter memory N	R/W	0x0000_0000_0000_0000	

14.4 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host has determined that a security operation is required, it creates a descriptor containing all the information the SEC needs to perform the security operation. The host creates the descriptor in main memory, then writes a pointer to the descriptor into the fetch FIFO of one of the SEC channels. The channel uses this pointer to read the descriptor into its descriptor buffer. Once it obtains the descriptor, the SEC uses its bus mastering capability to obtain inputs and write results, thus off-loading data movement and encryption operations from the host processor.

For test purposes, it is also possible for the host to write keys, context, and text-data directly to the SEC, using SEC's host-controlled mode. This method avoids use of descriptors.

14.4.1 Descriptor Structure

SEC descriptors are conceptually similar to descriptors used by most devices with DMA capability. The descriptors have a fixed length of 64 bytes, i.e. eight 64-bit words (referred to as dwords). A descriptor consists of one header dword and seven pointer dwords, as shown in Figure 14-3.

The header dword specifies the security operation to be performed, the execution unit(s) needed, and the modes for each execution unit. The pointer dwords, all of which have the same format, contain pointer and

	0–15	16	17–23	24–31	32–63
Header Dword	Header				Reserved
Pointer Dword 0	Length0	J0	Extent0	—	Pointer0
Pointer Dword 1	Length1	J1	Extent1	—	Pointer1
Pointer Dword 2	Length2	J2	Extent2	—	Pointer2
Pointer Dword 3	Length3	J3	Extent3	—	Pointer3
Pointer Dword 4	Length4	J4	Extent4	—	Pointer4
Pointer Dword 5	Length5	J5	Extent5	—	Pointer5
Pointer Dword 6	Length6	J6	Extent6	—	Pointer6

Figure 14-3. Descriptor Format

length information for locating input or output data parcels (such as keys, context, or text-data). The large number of pointers provided in the descriptor allows for multi-algorithm operations that require fetching of multiple keys, as well as fetch and return of contexts. Any pointer double word that is not needed can be given a length of zero and the channel skips over the corresponding operations.

SEC descriptors include scatter/gather capability, which means that each pointer in a descriptor can be either a direct pointer to a contiguous parcel of data, or can be a pointer to a ‘link table’ which is a list of pointers and lengths used to assemble the data parcel. When a link table is used to read input data, this is referred to as a gather operation; when used to write output data, it is referred to as a scatter operation.

14.4.2 Descriptor Format—Header Dword

Descriptors are created by the host to guide the SEC through required cryptographic operations. The descriptor header defines the operations to be performed, the mode for each operation, and internal addressing used by the controller and channel for internal data movement. The SEC device drivers allow the host to create proper headers for each cryptographic operation.

	0–3	4–11	12–15	16–23	24–28	29	30	31
Field	OP_0		OP_1		DESC_TYPE	—	DIR	DN
	EU_SEL0	MODE0	EU_SEL1	MODE1				

	32–63
Field	—

Figure 14-4. Header Dword

Table 14-4. Header Dword Bit Definitions

Bits	Name	Description
OP_0		
0–3	EU_SEL0	Primary EU select: See 14.4.2.1/14-17 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 14.4.2.1/14-17 for possible values.
16–23	MODE1	Secondary mode: Mode data used to program the primary EU. The mode data is to the chosen EU. This field is passed directly to bits 56–63 of the mode register in the selected EU.
24–28	DESC_TYPE	Descriptor type. This field, along with the DIR field, determines the sequence of actions to be performed by the channel and selected EUs using the blocks of data listed in the rest of the descriptor. The attributes determined include the direction of data flow for each data block, which EU (primary or secondary) is accessed, what snooping options are used, and which internal EU addresses are accessed. See 14.4.2.2/14-18 for possible values.
29	—	Reserved.
30	DIR	Direction of overall data flow 0 Outbound 1 Inbound This field, along with the DESC_TYPE field, helps determine the sequence of actions to be performed by the channel and selected EUs.
31	DN	Done notification 0 No done notification. 1 Signal done to the host on completion of this descriptor. The done notification can take the form of an interrupt, or a modified header writeback, or both, depending upon the states of the channel done interrupt enable (CDIE) and channel done writeback enable (CDWE) bits in the channel configuration register. When writeback is used, the value written back is 0xFF in bits 0–7 of this long word.

14.4.2.1 Selecting Execution Units—EU_SEL0 and EU_SEL1

Table 14-5 shows the values for EU_SEL0 and EU_SEL1 in the descriptor header. The following rules govern the choices for these fields:

1. EU_SEL0 values of 0000 (no EU selected) or 0111–1111 (reserved) results in an unrecognized header error condition during processing of the descriptor header.
2. The only valid choices for EU_SEL1 are 0000 (No EU selected) or 0011 (MDEU). Any other choice causes an unrecognized header error condition.
3. If EU_SEL1 is 0011 (MDEU), then EU_SEL0 must be 0010 (DEU), 0110 (AESU), or 0001 (AFEU). All other values of EU_SEL0 causes an unrecognized header error condition.

Table 14-5. EU_SEL1 and EU_SEL2 Values

Value (binary)	Selected EU
0000	No EU selected
0001	AFEU
0010	DEU
0011	MDEU
0100	RNG
0101	PKEU
0110	AESU
0111–1110	Reserved
1111	Reserved for header writeback

14.4.2.2 Selecting Descriptor Type—DESC_TYPE

Table 14-6 shows the permissible values for the DESC_TYPE field in the descriptor header. Descriptor types from the SEC 1.0, which have zero in the last bit (xxxx_0), are listed first, followed by new SEC 2.4 types, which have one in the last bit (xxxx_1).

Table 14-6. Descriptor Types

Value (binary)	Descriptor Type	Notes
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnooping [1]
0001_0	common_nonsnoop_no_afeu	Common, nonsnooping, non-PKEU, non-AFEU [1]
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU [2]
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_ 0	—	Reserved
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnooping
0001_0	common_nonsnoop	Common, nonsnooping, non-PKEU, non-AFEU
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_ 0	—	Reserved
0111_0	—	Reserved
1000_0	pkeu_mm	PKEU-Montgomery Multiplication

Table 14-6. Descriptor Types (continued)

Value (binary)	Descriptor Type	Notes
1001_0	—	Reserved
1010_0	—	Reserved
1011_ 0	—	Reserved
1100_0	hmac_snoop_aesu_ctr	AESU CTR hmac snooping [2]
1101_0	—	Reserved
1110_0	—	Reserved
1111_0	—	Reserved
0000_1	ipsec_esp	IPsec ESP mode encryption and hashing
0001_1	802.11i AES ccmp	CCMP encryption and hashing, suitable for 802.11i
0010_1	srtp	SRTP encryption and hashing
0011_1	pkeu_assemble	pkeu_assemble Elliptic Curve Cryptography
0100_1	pkeu_ptmul	pkeu_ptmul Elliptic Curve Cryptography
0101_1	pkeu_ptadd_dbl	pkeu_ptadd_dbl Elliptic Curve Cryptography
others	—	Reserved
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher
1001_1	tls_ssl_stream	TLS/SSL generic stream cipher
1010_1	raid_xor	XOR data streams
All others	—	Reserved

[1] Type 0000_0 is for AES-CTR operations. Type 0001_0 also supports AES-CTR; however, to use AES-CTR with 0001_0, the user must prepend zeros to the AES ctx before loading the AES context registers.

[2] Type 1100_0 is for AES-CTR operations with HMAC. Type 0010_0 also supports AES-CTR with HMAC; however, to use AES-CTR with 0010_0, the user must prepend zeros to the AES ctx before loading the AES context registers.

For more about descriptor types and the data used for each type, see Section 14.4.5, “Descriptor Types.”

14.4.3 Descriptor Format—Pointer Dwords

The descriptor contains seven pointer dwords that define where in memory the SEC should access its input and output data parcels. The channel determines how it will use each of the pointer dwords based on the descriptor type and direction fields in the header. The channel accesses the first data parcel by starting at a location given by a POINTER value, and accessing a number of bytes given by a LENGTH or EXTENT value. Subsequent data parcels may be accessed by staring where a previous data parcel ended, or by starting at a different POINTER. The LENGTH or EXTENT used with any POINTER may be from the same pointer dword or from a different pointer dword in the same descriptor. Although the EXTENT field

exists in each pointer dword of the SEC descriptor, (only the EXTENTs in pointer dwords 3, 4, and 5 are currently in use).

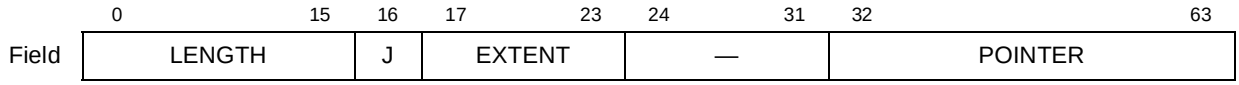


Figure 14-5. Pointer Dword

Table 14-7. Pointer Dword Field Definitions

Bits	Name	Description
0–15	LENGTH	Length. A number of bytes in the range 0 to 65535. The use of this field depends on the descriptor type and direction in the header dword. A value of zero causes the channel to skip this dword.
16	J	Jump. Determines whether to jump to a link table whenever the POINTER field in this same word is used. 0 The POINTER field points to data. 1 The POINTER field points to a link table, and scatter/gather is enabled.
17–23	EXTENT	Extent. A number of bytes in the range 0 to 127. The use of this field depends on the descriptor type and direction in the header dword.
24–31	—	Reserved
32–63	POINTER	Pointer. A memory address.

On occasion, a descriptor field may not be applicable to the requested service. With seven length/pointer pairs, it is possible that not all descriptor fields are required to load the required keys, context, and text-data. (Some operations, for example, do not require context.) Therefore, when processing descriptors the SEC skips entirely any pointer that has an associated LENGTH of zero.

Some descriptors involve more than seven parcels of input and output data. In these cases, it is necessary to use one POINTER field to address a sequence of data parcels.

LENGTH and EXTENT fields normally specify the sizes of data parcels. In some cases, however, the POINTER field is zero, and the LENGTH and/or EXTENT fields simply specify values to be written to an EU.

If the channel procedure calls for reading a parcel using a nonzero LENGTH field, but the POINTER field is zero, the length value is written to the EU but no data parcel is fetched from the bus.

The J bit in each pointer dword is used to enable the scatter/gather feature. If a data parcel to be read or written by SEC is in one contiguous block of memory locations, then the scatter/gather feature is not needed. In this case the POINTER should be set to point directly at the first byte of the parcel, and the J bit should be 0. On the other hand, if the data parcel is stored in several separate segments of memory, then the scatter/gather capability is needed to assemble or distribute the complete parcel. In this case the POINTER should be set to point to a link table, and the J bit should be 1. For link table format, see Section 14.4.4, “Link Table Format.”

Scatter/gather capability is available for all pointer dwords of all descriptor types, with the following exception(s):

- Raid-XOR descriptor type does not allow scatter/gather.

14.4.4 Link Table Format

Link tables implement scatter/gather capability. For gather operations, a link table specifies a list of memory segments that are to be concatenated in the process of assembling data parcels. For scatter operations, a link table specifies a list of memory segments into which the output data should be written. Scatter or gather of a data parcel may be specified by a single link table or by a chain of link tables that are linked together with pointers (see Figure 14-7).

The link table or chain of link tables accessed through some descriptor POINTER must specify enough memory segments to hold all the data that will be accessed through that pointer. In most cases, only a single data parcel is accessed through a given POINTER, and the chain of link tables specifies just that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of data parcels, and the chain of link tables must supply data for the entire sequence. If a link table is used to access a sequence of data parcels, the end of each parcel must also be at the end of a memory segment. In other words, a single memory segment must not straddle two data parcels. An example of proper construction of link tables is illustrated in Figure 14-7.

A link table may contain any number of long word entries. There are two kinds of entries, ‘regular’ entries and ‘next’ entries. Each ‘regular’ entry specifies a memory segment by means of a 32-bit starting address (SEGPTR) and a 16-bit length (SEGLEN). A ‘next’ entry is used at the end of a link table to specify that the list of memory segments is continued in another link table. In a ‘next’ entry, the N bit is set and the SEGPTR field gives the address of the next link table, and the SEGLEN field must be 0. A chain of link tables may contain any number of link tables.

Whether the list of memory segments is in a single link table or split into several link tables, the last entry in the last link table is a ‘regular’ entry with the R (return) bit set. The R bit signifies the end of link table operations so that the channel returns to the descriptor for its next pointer (if any). Link tables are illustrated in Figure 14-7.

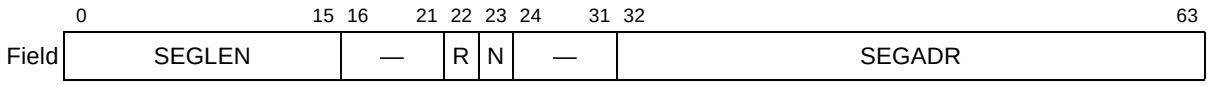


Figure 14-6. Link Table Entry Format

Table 14-8. Link Table Field Definitions

Bits	Name	Description
0–15	SEGLEN	Length. When N=0, a number in the range 1 to 65535, specifying the number of bytes in the memory segment. pointed to by SEGADR. A value of 0 will cause an error bit to be set in the Channel Pointer Status Register—GER for a gather operation or SER for a scatter operation (see 14.6.1.1/14-82). When N=1, must be zero.
16–21	—	Reserved
22	R	Return: When N=0: 0 No special action. 1 This is the last entry in the chain of link tables. If this entry does not specify the right number of bytes to complete the last data parcel, a GER or SER error will be set in the Channel Pointer Status Register (see 14.6.1.1/14-82). When N=1: ignored.

Table 14-8. Link Table Field Definitions (continued)

Bits	Name	Description
23	N	Next 0 No special action. 1 This is the last long word in the current link table. The SEGADR field is the address of the next link table in the chain.
24–31	—	Reserved
32–63	SEGADR	Segment address. A memory address.

For any sequence of data parcels accessed by a link table or chain of link tables, the combined lengths of the parcels (the sum of their LENGTH and/or EXTENT fields) must equal the combined lengths of the link table memory segments (SEGLEN fields). Otherwise the channel sets the appropriate error bit in the channel pointer status register—GER for a gather error or SER for a scatter error. See Section 14.6.1.1, “Crypto-Channel Configuration Register (CCCR),” for more information.

14.4.4.1 Link Table Example

Figure 14-7 is an example of proper construction of link tables.

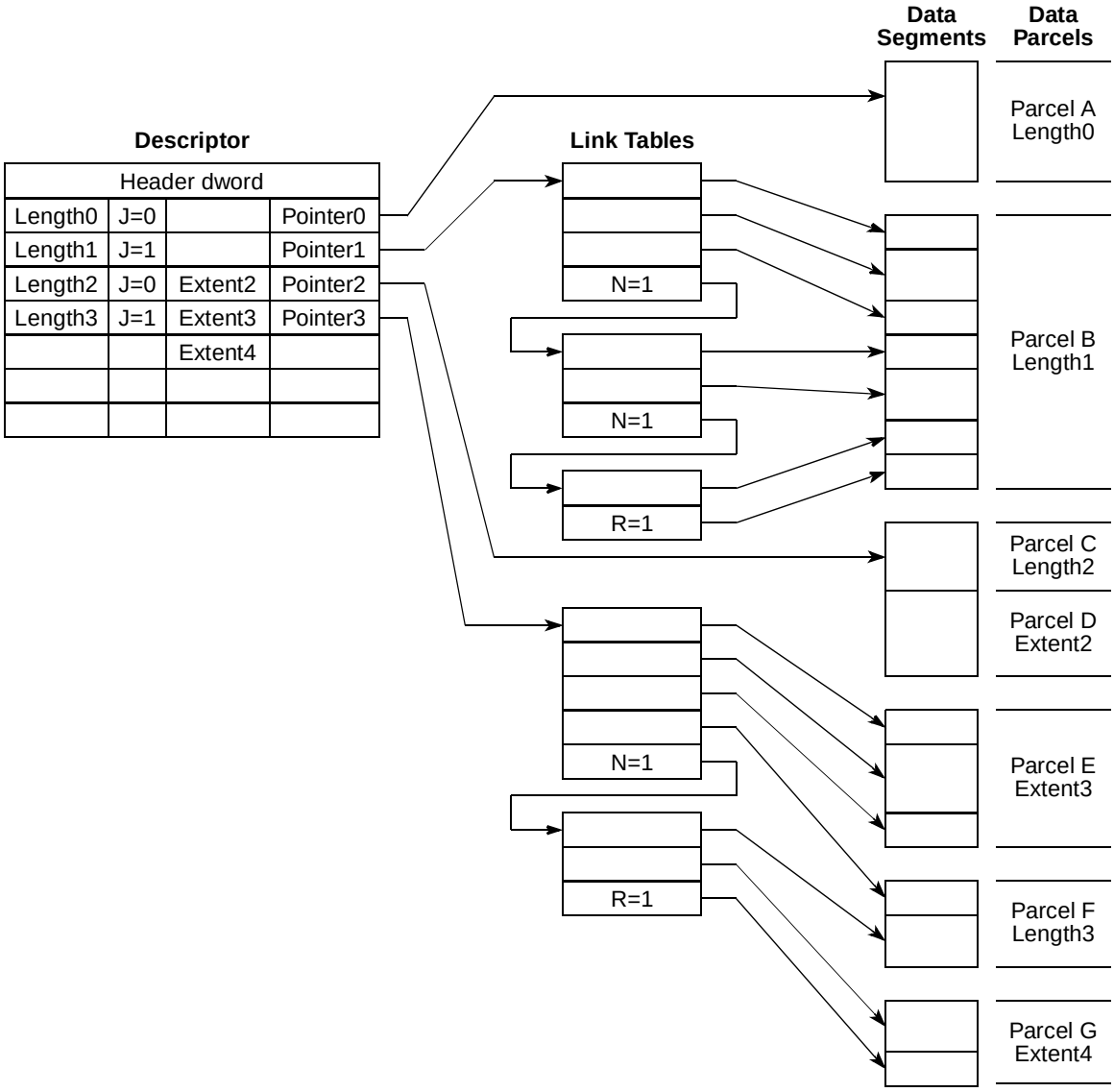


Figure 14-7. Descriptors, Link Tables, and Data Parcels

Figure 14-7 illustrates various ways that a descriptor may specify data parcels:

- The first pointer double word in the descriptor specifies Parcel A using the simplest method: the parcel is specified directly through Pointer0 and Length0.
- The next pointer double word uses a chain of link tables to specify Parcel B. Since J=1, pointer1 is used as the address of a link table. The link table specifies several ‘regular’ entries specifying data segments to be concatenated. The last word of the link table is a next entry indicating that the list continues in the ‘next’ link table. The last entry in the last link table of the chain has the R bit set.
- The last cases illustrate how one pointer in a descriptor can be used to specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterwards, with length

specified by Extent2. Pointer3 is used for three data parcels (E, F, and G), this time using link tables.

To demonstrate use of a link table, assume that the current descriptor type calls for the channel to access a data parcel using Pointer3 and Extent3 fields, and assume that J3=1. Due to the J3 value, Pointer3 is not used as a data address but instead used as the address of a link table. The channel begins by reading the first four long words starting at Pointer3 into an internal link table buffer.

Using the first entry of the link table, the channel starts accessing the data parcel by reading SEGLEN bytes beginning at SEGADR. If the required data parcel size (Extent3) is greater than this first SegLen, the channel moves on to the next entry of the link table, and reads SEGLEN bytes starting at SEGADR. While there are more bytes to be read in the data parcel, this process continues. If the channel's link table buffer is exhausted, the channel reads the next four long words of the link table into its link table buffer. If a link table entry is encountered in which the N bit is set, the channel uses the SEGADR field in that word to find the next link table in the chain. The last byte of the required parcel size (Extent3) must coincide with the last byte of a memory segment, or unpredictable results may occur.

Now assume that the channel accesses its next data parcel using Pointer3 again, this time with length given by Length3. In this case the channel continues to the next line of the link table, and begins reading the memory segment specified there. As before, the channel concatenates memory segments from as many link table entries as necessary to obtain the required number of bytes (Length3).

Similarly, the next data parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 data parcel is the last one to be accessed through Pointer3. Then the link table entry that supplies the last memory segment for Extent4 has the R bit set, signifying that this is the last entry in the chain of link tables.

14.4.5 Descriptor Types

Table 14-9 shows how the pointer dwords should be used with the various descriptor types to load keys, context, and text data into the execution units, and how the required outputs should be unloaded. Additional explanation of the use of certain descriptor types and the meaning of the pointer dwords can be found in the *SEC 2.4 Descriptor Programmer's Guide*.

Table 14-9. Descriptor Pointer Dword Usage

Descriptor Type	Pointer DWORD 1	Pointer DWORD 2	Pointer DWORD 3	Pointer DWORD 4	EXTENT 4	Pointer DWORD 5	EXTENT 5	Pointer DWORD 6	EXTENT 6	Pointer DWORD 7
0000_0	nil	Cipher IV	Cipher Key	In FIFO	nil	Out FIFO	nil	Cipher IV Out	nil	nil
0001_0	nil	Cipher IV	Cipher Key	In FIFO	nil	Out FIFO	nil	Cipher IV Out	nil	nil
0010_0	HMAC Key	HMAC Data	Cipher Key	Cipher IV	nil	In FIFO	nil	Out FIFO	nil	HMAC Out
0011_0	Reserved									
0100_0	Reserved									
0101_0	nil	ARC-4 Context (In FIFO)	ARC-4 Key	In FIFO	nil	Out FIFO	nil	ARC-4 Context (Out FIFO)	nil	nil
0110_ 0	Reserved									
0111_0	Reserved									
1000_0	N	B	A	E	nil	B Out	nil	nil	nil	nil
1001_0	Reserved									
1010_0	Reserved									
1011_ 0	Reserved									
1100_0	HMAC Key	HMAC Data	AES Key	AES Ctx	nil	In FIFO	nil	Out FIFO	nil	HMAC Out
1101_0	Reserved									
1110_0	Reserved									
1111_0	Reserved									
0000_1	HMAC Key	HMAC Data	Cipher IV	Cipher Key	nil	In FIFO	nil	Out FIFO	HMAC Out	Cipher IV Out
0001_1	nil	AES-Ctx	AES Key	In FIFO	nil	In FIFO	nil	Out FIFO	nil	AES-Ctx-Out
0010_1	HMAC Key	AES-Ctx	AES Key	In FIFO	In FIFO	Out FIFO	In FIFO	HMAC Out	nil	AES-Ctx-Out
0011_1	A0	A1	A2	A3	nil	B0	nil	B1	nil	'Build'
0100_1	N	E	'Build'	B1 Out	nil	B2 Out	nil	B3 Out	nil	nil
0101_1	N	'Build'	B2	B3	nil	B1 Out	nil	B2 Out	nil	B3 Out
1000_1 Outbound	MAC Key	Cipher IV	Cipher Key	In FIFO	In FIFO	In FIFO	MAC In	Out FIFO	nil	Cipher IV Out
1000_1 Inbound	MAC Key	Cipher IV	Cipher Key	nil	In FIFO	In FIFO	MAC In	Out FIFO	MAC Out	Cipher IV Out
1001_1 Outbound	MAC Key	Cipher IV	Cipher Key	In FIFO	In FIFO	In FIFO	MAC Out	Out FIFO	nil	Cipher IV Out
1001_1 Inbound	MAC Key	Cipher IV	Cipher Key	nil	In FIFO	In FIFO	MAC In	Out FIFO	MAC Out	Cipher IV Out
1010_1	nil	nil	nil	In 1	undefined	In 2	undefined	In 3	undefined	Out
others	Reserved									

14.5 Execution Units

Execution unit (EU) is the term used for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i processing, and can work together to perform high level cryptographic tasks.

The following execution units are used in the SEC:

- Public key execution unit (PKEU)
- Data encryption standard execution unit (DEU)
- Advanced encryption standard execution unit (AESU) implementing the Rijndael symmetric key cipher.
- ARC four execution unit (AFEU)
- Message digest execution unit (MDEU)
- One private on-chip random number generator (RNG)

Working together, the EUs can perform high-level cryptographic tasks, such as IPSec Encapsulating Security Protocol (ESP) and digital signature. The remainder of this chapter provides details about the execution units themselves.

14.5.1 Public Key Execution Unit (PKEU)

This section contains details about the public key execution unit (PKEU), including detailed register map, modes of operation, status and control registers, and the parameter RAMs. The following sections describe PKEU registers and parameter memories.

14.5.1.1 PKEU Mode Register (PKEUMR)

PKEUMR specifies the internal PKEU routine to be executed. The mode register is cleared when the PKEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error will be generated.

Figure 14-8 shows the PKEU Mode Register, and Table 14-10 lists the possible mode field values.

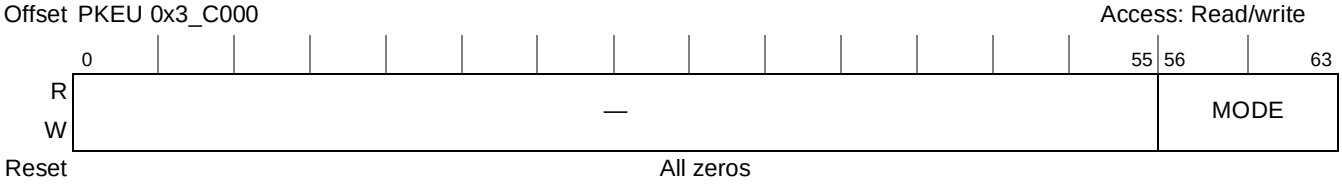


Figure 14-8. PKEU Mode Register

Table 14-10 lists the possible mode field values. Depending on the value written to PKEU[Mode] field depends on the routine used. Parameter memories are referred to for the base address, as shown.

Table 14-10. Mode Field Description

Routine Name	Routine Description	Mode [56–63]
RESERVED	Reserved	0x00
CLEARMEMORY	Clear memory	0x01
MOD_EXP	FP: Exponentiate mod N and deconvert from Montgomery format	0x02
MOD_R2MODN	FP: Compute Montgomery converter (R2 mod N)	0x03
MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem (RnRp mod N)	0x04
EC_FP_AFF_PTMULT	FP EC: Multiply key times point in affine coordinates	0x05
EC_F2M_AFF_PTMULT	F2m EC: Multiply key times point in affine coordinates	0x06
EC_FP_PROJ_PTMULT	FP EC: Multiply key times point in projective coordinates	0x07
EC_F2M_PROJ_PTMULT	F2m EC: Multiply key times point in projective coordinates	0x08
EC_FP_ADD	FP EC: Add two points in projective coordinates	0x09
EC_FP_DOUBLE	FP EC: Double a point in projective coordinates	0x0A
EC_F2M_ADD	F2m EC: Add two points in projective coordinates	0x0B
EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates	0x0C
F2M_R2	F2m: Compute Montgomery converter (R2 mod N)	0x0D
F2M_INV[1]	F2m: Invert mod N	0x0E
MOD_INV[2]	FP: Invert mod N	0x0F
MOD_ADD	FP: Add mod N	0x10
MOD_SUB	FP: Subtract mod N	0x20
MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format	0x30
MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format	0x40
F2M_ADD	F2m: Add mod N	0x50
F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format	0x60
F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format	0x70
RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, F2M_MULT1_MONT, and MOD_EXP)	0x80
SPK_BUILD	Build PK data structure	0xFF

1 F2M_INV returns incorrect results for modulus sizes greater than 480 bits.

2 MOD_INV returns incorrect results for modulus sizes greater than 480 bits.

14.5.1.2 PKEU Key Size Register (PKEUKSR)

The PKEU key size register, shown in Figure 14-9, reflects the number of significant bytes to be used from PKEU parameter memory E in performing modular exponentiation or elliptic curve point multiplication. Note that leading zeros are note significant and are not considered part of the key (modulus) size. The range of values for this register, when performing either modular exponentiation or elliptic curve point multiplication, is from 1 to 256. Specifying a key size outside of this range will cause a key size error in the PKEU interrupt status register (PKEUISR[KSE] is set).

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

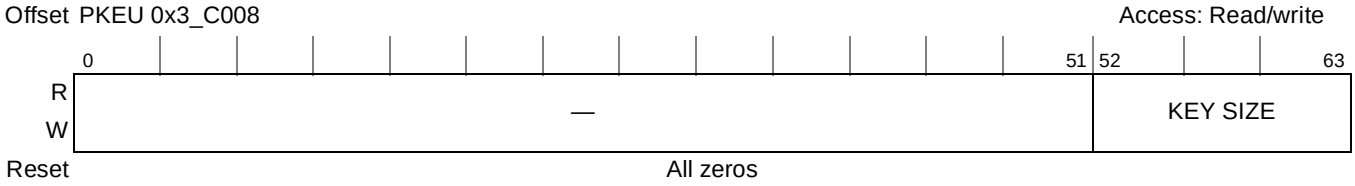


Figure 14-9. PKEU Key Size Register

14.5.1.3 PKEU AB Size Register (PKEUABS)

PKEU ABS (Figure 14-10) represents the operand size for the specific operands whenever it is required. The unit of the value written into PKEUABS[AB Size] is in bits, even though internally the PKEU imposes a 32-bit alignment. Any data beyond the number of bits in PKEUABS, either in A- and B-ram (operands) will be ignored. No error checking is performed whether the operand sizes are greater than the prime modulus or the field size and this may cause a wrong result. In other words, it is assumed that operands are modulo reduced before being written into the PKEU. Hence, PKEUABS[AB Size] must be less than or equal to data size for a correct result. If PKEUABS is modified during processing, an error will be generated.

An illegal data size error is generated as follows:

- For all non-ECC routines, a data size > 256 generates an illegal data size error.
- For all ECC routines, a data size > 64 generates an illegal data size error.

Setting PKEUABS[AB Size] = 0 (either intentionally or by ignoring it and not writing it at all) generates an illegal size error, except for routines such as the CLEAR_MEM routine that do not require an A or B operand.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

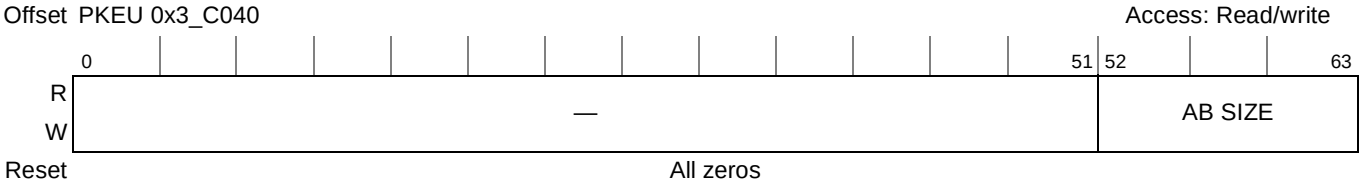


Figure 14-10. PKEU AB Size Register

14.5.1.4 PKEU Data Size Register (PKEUDSR)

PKEUDSR, shown in Figure 14-11, specifies the size in bits of the significant portion of the modulus or irreducible polynomial. Any value written to this register that is a multiple of 32 bits (for example, 128 bits or 160 bits) will be represented internally as the same value (128 bits or 160 bits). Any value written that is not a multiple of 32 bits (for example, 132 bits or 161 bits) will be represented internally as the next larger 32 bit multiple (160 bits or 196 bits). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 97 bits (internally 128 bits). The maximum size to operate properly is 2048 bits. A value in bits larger than 2048 results in a data size error.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register prior to performing debug operations.

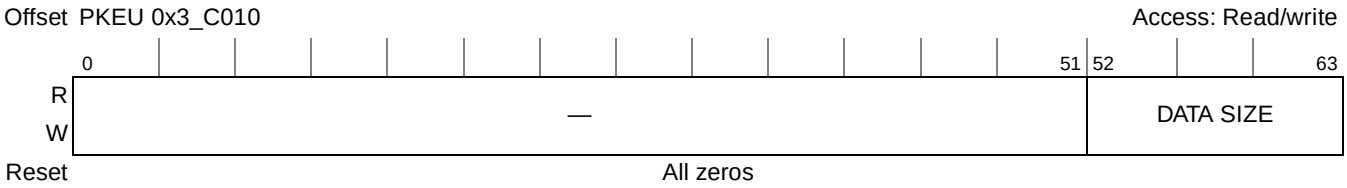


Figure 14-11. PKEU Data Size Register (PKEUDSR)

14.5.1.5 PKEU Reset Control Register (PKEURCR)

PKEURCR, shown in Figure 14-12, contains three reset options specific to the PKEU.

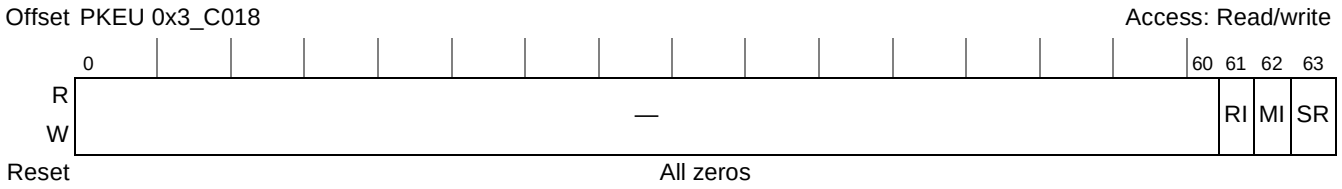


Figure 14-12. PKEU Reset Control Register (PKEURCR)

Table 14-11 describes PKEURCR fields.

Table 14-11. PKEURCR Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes PKEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the PKEU interrupt status register. 0 Do not reset interrupt logic. 1 Reset interrupt logic.
62	MI	Module initialization. Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RD (reset done) bit in the PKEU status register (Section 14.5.1.6, "PKEU Status Register (PKEUSR)"). 0 Do not reset most of PKEU. 1 Reset most of PKEU.
63	SR	SW reset. Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the PKEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the PKEU will enter a routine to perform proper initialization of the parameter memories. The RD (reset done) bit in the PKEU status register will indicate when this initialization routine is complete. (See Section 14.5.1.6, "PKEU Status Register (PKEUSR).") 0 Do not reset full PKEU. 1 Full PKEU reset.

14.5.1.6 PKEU Status Register (PKEUSR)

PKEUSR fields reflect the state of PKEU internal fields. PKEUSR is read only. Writing to it results in an address error being reflected in the PKEU interrupt status register (PKEUISR[AE] is set).

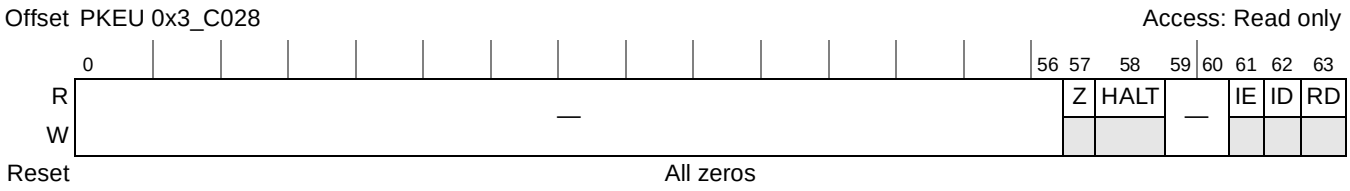


Figure 14-13. PKEU Status Register (PKEUSR)

Table 14-12 describes PKEUSR fields.

Table 14-12. PKEU Status Register Field Descriptions

Bits	Name	Description
0–56	—	Reserved
57	Z	Zero. Reflects the state of the PKEU zero detect bit when last sampled. Only particular instructions within routines cause zero to be modified, so this bit should be used with great care.
58	HALT	Halt indicates whether the PKEU has halted due to an error. 0 PKEU not halted 1 PKEU halted **Note:** Because the error causing the PKEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.

Table 14-12. PKEU Status Register Field Descriptions (continued)

Bits	Name	Description
59–60	—	Reserved
61	IE	Interrupt error. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (see Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling error 1 PKEU is signaling error
62	ID	Interrupt done. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (see Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 PKEU is not signaling done 1 KEU is signaling done
63	RD	Reset done. 0 Reset in progress 1 Reset done. PKEU has completed its internal reset sequence.

14.5.1.7 PKEU Interrupt Status Register (PKEUISR)

The PKEU interrupt status register tracks the state of possible errors, if those errors are not masked, through the PKEU interrupt control register (PKEUICR). The definition of each bit in the PKEUISR is shown in Figure 14-14.

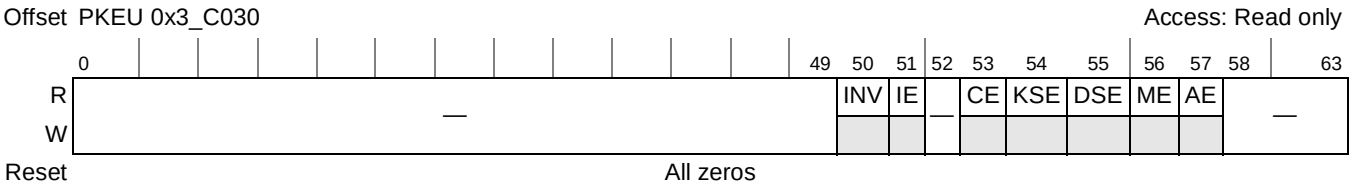


Figure 14-14. PKEU Interrupt Status Register (PKEUISR)

Table 14-13 describes PKEUISR fields.

Table 14-13. PKEUISR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error. Indicates that the inversion routine has a zero operand. 0 No inversion error detected 1 Inversion error detected
51	IE	Internal error. An internal processing error was detected while the PKEU was operating. 0 No error detected 1 Internal error **Note:** This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the PKEU.
52	—	Reserved
53	CE	Context error. A PKEU key register, the key size register, the data size register, or mode register was modified while the PKEU was operating. 0 No error detected 1 Context error

Table 14-13. PKEUISR Field Descriptions (continued)

Bits	Name	Description
54	KSE	Key size error. Value outside the bounds of 1–256 bytes was written to the PKEU key size register 0 No error detected 1 Key size error detected
55	DSE	Data size error. Value outside the bounds 97– 2048 bits was written to the PKEU data size register 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in the mode register. 0 No error detected 1 Mode error **Note:** Writing to reserved bits in a mode register is a likely source of error.
57	AE	Address error. Illegal read or write address was detected within the PKEU address space. 0 No error detected 1 Address error
58–63	—	Reserved

14.5.1.8 PKEU Interrupt Control Register (PKEUICR)

The PKEU interrupt control register controls the result of detected errors. For a given error (as defined in Section 14.5.1.7, “PKEU Interrupt Status Register (PKEUISR)”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, PKEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

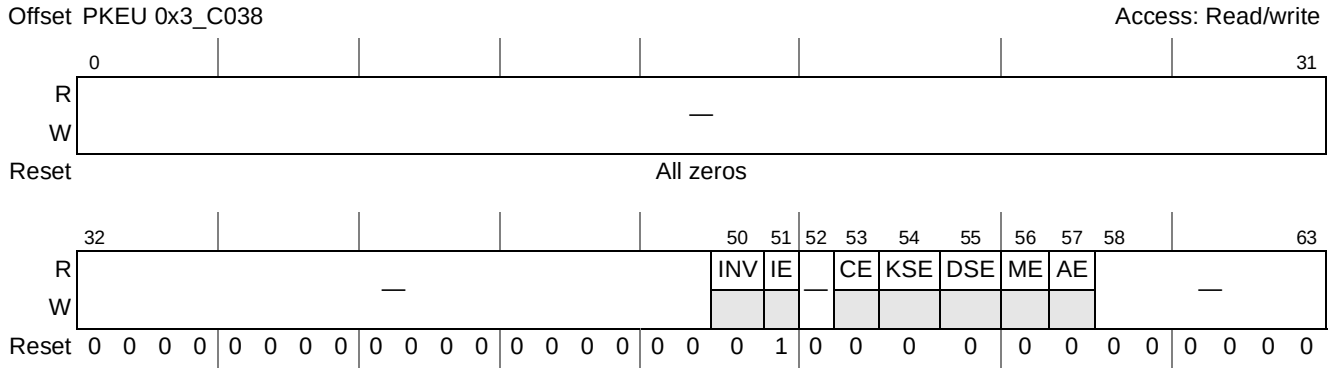


Figure 14-15. PKEU Interrupt Control Register (PKEUICR)

Table 14-14 describes PKEUICR fields.

Table 14-14. PKEUICR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	INV	Inversion error 0 Inversion error enabled 1 Inversion error disabled
51	IE	Internal error 0 Internal error enabled 1 Internal error disabled
52	—	Reserved
53	CE	Context error 0 Context error enabled 1 Context error disabled
54	KSE	Key size error 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error 0 Mode error enabled 1 Mode error disabled
57	AE	Address error 0 Address error enabled 1 Address error disabled
58–63	—	Reserved

14.5.1.9 PKEU EU-Go Register (PKEUEUG)

The EU-Go register in the PKEU is used to indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the mode register, per the contents of the parameter memories listed below. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. PKEUEUG is only used when the is operated as a slave. The descriptors and crypto-channel activate the PKEU (through an internally generated write to PKEUEUG) when the acts as an initiator.

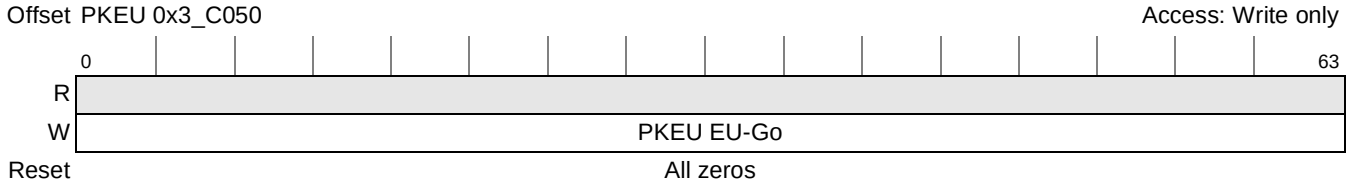


Figure 14-16. PKEU EU-Go Register (PKEUEUG)

14.5.1.10 PKEU Parameter Memories

The PKEU uses four 2048-bit memories to receive and store operands for the arithmetic operations the PKEU will be asked to perform. In addition, results are stored in one particular parameter memory.

All these memories store data in the same format: least significant data byte in the least significantly addressed byte, both data significance and addressing significance increasing identically and simultaneously.

14.5.1.10.1 PKEU Parameter Memory A

This 2048-bit memory is typically used as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 512-bit memories, and is used to specify particular curve parameters and input values.

14.5.1.10.2 PKEU Parameter Memory B

This 2048-bit memory is typically used as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory serves as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 512-bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.

14.5.1.10.3 PKEU Parameter Memory E

This 2048-bit memory is non-segmentable, and stores the exponent for modular exponentiation, or the multiplier k for elliptic curve point multiplication. This memory space is write only; a read of this memory space will cause address error to be reflected in the PKEUISR.

14.5.1.10.4 PKEU Parameter Memory N

This 2048-bit memory is non-segmentable, and stores the modulus for modular arithmetic and F_p elliptic curve routines. For F_2M elliptic curve routines, this memory stores the irreducible polynomial.

14.5.2 Data Encryption Standard Execution Unit (DEU)

This section contains details about the data encryption standard execution unit (DEU), including detailed register map, modes of operation, status and control registers, and FIFOs.

The registers used in the DEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the DEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and on-chip controller will abstract register-level access from the user.

14.5.2.1 DEU Mode Register (DEUMR)

The DEU mode register contains 3 bits that are used to program the DEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the SEC as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

DEUMR is cleared when the DEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error will be generated.

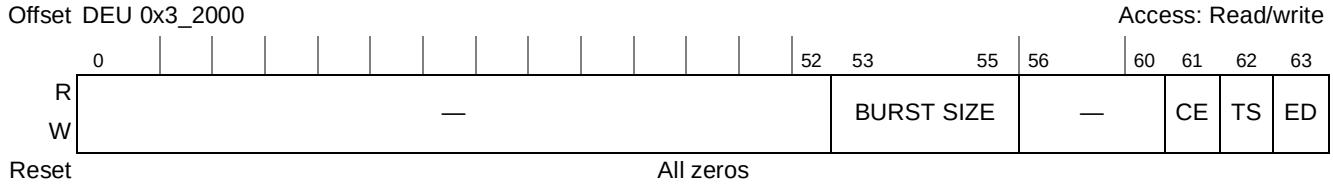


Figure 14-17. DEU Mode Register (DEUMR)

Table 14-15 describes DEU mode register fields.

Table 14-15. DEUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	Implements flow control to allow larger than FIFO-sized blocks of data to be processed with a single key/IV. The DEU signals to the channel that a BURST SIZE amount of data is available to be pushed to or pulled from the FIFO. **Note:** This field is included to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the DEU.
56–60	—	Reserved
61	CE	CBC/ECB. If set, DEU operates in cipher-block-chaining mode. If not set, DEU operates in electronic codebook mode. 0 ECB mode 1 CBC mode
62	TS	Triple/Single DES. If set, DEU performs the triple DES algorithm; if not set, DEU performs the single DES algorithm. 0 Perform single DES. 1 Perform triple DES.
63	ED	Encrypt/Decrypt. If set, DEU performs the encryption algorithm; if not set, DEU performs the decryption algorithm. 0 Perform decryption. 1 Perform encryption.

14.5.2.2 DEU Key Size Register (DEUKSR)

The DEUKSR indicates the number of bytes of key memory used in encrypting or decrypting. If DEUMR is set for single DES, any value other than 8 bytes automatically generates a key size error in the DEUISR. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1=K3) or 24 bytes (168 bits for 3-key triple DES) generates an error. Triple DES always uses K1 to encrypt, K2 to decrypt, K3 to encrypt.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

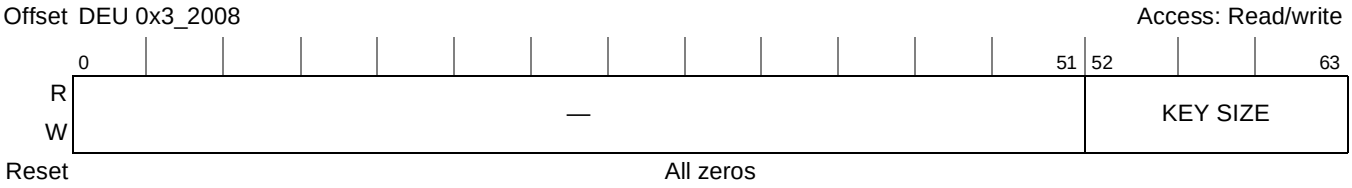


Figure 14-18. DEU Key Size Register (DEUKSR)

Table 14-16 shows the legal values for DEU key size.

Table 14-16. DEUKSR Field Descriptions

Bits	Name	Description
0–51	—	Reserved
52–63	Key Size	8 bytes = 0x08 (only legal value if mode is single DES.) 16 bytes= 0x10 (for 2 key 3DES, K1 = K3) 24 bytes= 0x18 (for 3 key 3DES)

14.5.2.3 DEU Data Size Register (DEUDSR)

The DEUDSR, shown in Figure 14-19, is used to verify that the data to be processed by the DEU is divisible by the DES algorithm block size of 64 bits. The DEU does not automatically pad messages out to 64-bit blocks; therefore, any message processed by the DEU must be divisible by 64 bits or a data size error will occur.

In normal operation, the full message length (data size) to be encrypted or decrypted by the DEU is copied from the descriptor to DEUDSR; however, only bits 58–63 are checked to determine if there is a data size error. If bits 58–63 are all zeros, the message is evenly divisible into 64-bit blocks. In target mode, the user must write the data size to DEUDSR. If the data size written is not divisible by 64-bits (bits 58–63 nonzero), a data size error will occur.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

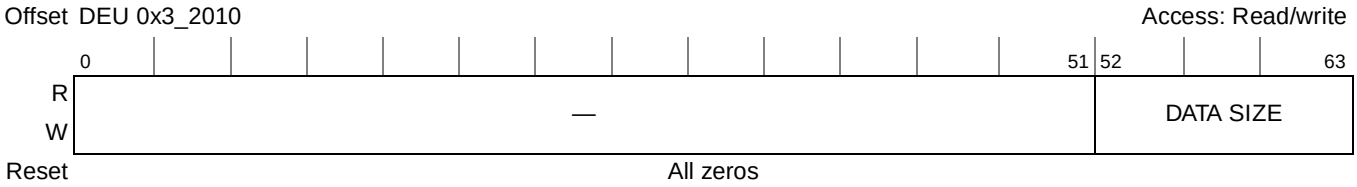


Figure 14-19. DEU Data Size Register (DEUDSR)

14.5.2.4 DEU Reset Control Register (DEURCR)

The DEURCR, shown in Figure 14-20, allows three levels reset of just DEU, as defined by the three self-clearing bits:

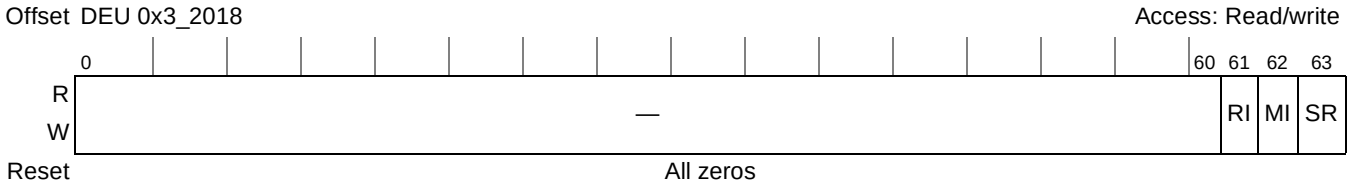


Figure 14-20. DEU Reset Control Register (DEURCR)

Table 14-17 describes DEURCR fields.

Table 14-17. DEURCR Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes DEU interrupts signaling DONE and ERROR to be reset. It further resets the state of DEUISR. 0 Don't reset. 1 Reset interrupt logic.
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in DEUSR. 0 Don't reset. 1 Reset most of DEU.
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in DEUSR will indicate when this initialization routine is complete 0 Don't reset. 1 Full DEU reset

14.5.2.5 DEU Status Register (DEUSR)

The DEU status register, shown in Figure 14-21, contains six fields which reflect the state of DEU internal signals. DEUSR is read only; writing to DEUSR causes an address error being reflected in DEUISR.

Offset DEU 0x3_2028 Access: Read-only

	0–39	40–47	48–55	56–57	58	59–60	61	62	63
R	—	OFL	IFL	—	HALT	—	IE	ID	RD
W									
Reset	All zeros								

Figure 14-21. DEU Status Register (DEUSR)

Table 14-12 describes DEUSR fields.

Table 14-18. DEUSR Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the DEU has halted due to an error. 0 DEU not halted 1 DEU halted **Note:** Because the error causing the DEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	IE	Interrupt error. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 DEU is not signaling error 1 DEU is signaling error
62	ID	Interrupt done. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 DEU is not signaling done 1 DEU is signaling done
63	RD	Reset done. 0 Reset in progress 1 Reset done. DEU has completed its internal reset sequence.

14.5.2.6 DEU Interrupt Status Register (DEUISR)

The DEU interrupt status register, shown in Figure 14-22, tracks the state of possible errors, if those errors are not masked, via the DEU interrupt control register.

Offset DEU 0x3_2030 Access: Read only

	0–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
W															
Reset	All zeros														

Figure 14-22. DEU Interrupt Status Register (DEUISR)

Table 14-19 describes DEUISR fields.

Table 14-19. DEUISR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. Defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are checked for parity only if the appropriate DEU mode register bit indicates triple DES. Also, key register 3 is checked only if key size reg = 24. Key register 2 is checked only if key size reg = 16 or 24.) 0 No error detected 1 Key parity error
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error **Note:** IE is asserted anytime an enabled error condition occurs and can only be cleared by setting the corresponding bit in the DEUICR or by resetting the DEU.
52	ERE	Early read error. The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. A DEU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while DEU was performing encryption. 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU key size register 0 No error detected 1 Key size error
55	DSE	Data size error: A value was written to the DEU Data Size Register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	IFU	Input FIFO underflow. The DEU input FIFO has been read while empty. 0 No error detected 1 Input FIFO has had underflow error

Table 14-19. DEUISR Field Descriptions (continued)

Bits	Name	Description
61	IFO	Input FIFO overflow. The DEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed **Note:** When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	OFO	Output FIFO overflow. The DEU output FIFO has been pushed while full. 0 No error detected 1 Output FIFO has overflowed

14.5.2.7 DEU Interrupt Control Register (DEUICR)

The DEU interrupt control register controls the result of detected errors. For a given error (as defined in Section 14.5.2.6, “DEU Interrupt Status Register (DEUISR)”),if the corresponding bit in this register is set, then the error is ignored, no error interrupt occurs and DEUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, DEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Offset PKEU 0x3_2038 Access: Read/write

	0–31
R	—
W	
Reset	All zeros

	32–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
W															
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-23. DEU Interrupt Control Register (DEUICR)

Table 14-20. DEUICR Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key parity error. The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity enabled 1 Key parity error disabled
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled

Table 14-20. DEUICR Field Descriptions (continued)

Bits	Name	Description
52	ERE	Early read error. The DEU IV Register was read while the DEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. A DEU key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for Triple DES) was written to the DEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error (DSE): A value was written to the DEU data size register that is not a multiple of 8 bytes. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of DONE interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	IFU	Input FIFO underflow. The DEU input FIFO has been read while empty. 0 Input FIFO Underflow error enabled 1 Input FIFO Underflow error disabled
61	IFO	Input FIFO overflow. The DEU input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled **Note:** When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	OFO	Output FIFO overflow. The DEU output FIFO has been pushed while full. 0 Output FIFO Overflow error enabled 1 Output FIFO Overflow error disabled

14.5.2.8 DEU EU-Go Register (DEUEUG)

The EU-Go register in the DEU is used to indicate a DES operation may be completed. After the final message block is written to the input FIFO, DEUEUG must be written. The value in the data size register

will be used to determine how many bits of the final message block (always 64) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to DEUEUG is merely a trigger causing the DEU to process the final block of a message, allowing it to signal DONE.

DEUEUG is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the DEU (through an internally generated write to DEUEUG) when the SEC acts as an initiator.

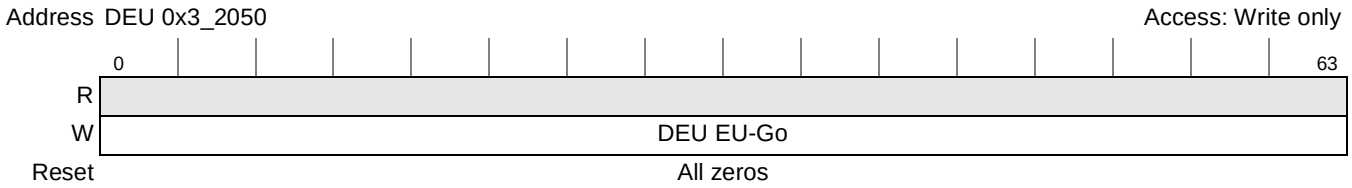


Figure 14-24. DEU EU-Go Register (DEUEUG)

14.5.2.9 DEU IV Register (DEUIV)

For CBC mode, the initialization vector is written to and read from DEUIV. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading DEUIV while the module is processing data generates an error interrupt.

14.5.2.10 DEU Key Registers (DEUK1–DEUK3)

The DEU uses three write-only key registers to perform encryption and decryption. In Single DES mode, only DEUK1 may be written. The value written to DEUK1 is simultaneously written to DEUK3, auto-enabling the DEU for 112-bit triple DES if DEUKSR indicates 2-key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit triple DES, DEUK1 must be written first, followed by DEUK2, then DEUK3.

Reading any of these memory locations generates an address error interrupt.

14.5.2.11 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the DEU FIFO address space causes the 64-bit-word to be pushed onto the DEU input FIFO, and a read from anywhere in the DEU FIFO address space causes a 64-bit-word to be popped off the DEU output FIFO. Overflows and underflows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

14.5.3 ARC Four Execution Unit (AFEU)

This section contains details about the ARC four execution unit (AFEU), including detailed register map, modes of operation, status and control registers, S-box memory, and FIFOs.

The registers used in the AFEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the AFEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.3.1 AFEU Mode Register (AFEUMR)

Shown in Figure 14-25, the AFEU mode register contains three bits which are used to program the AFEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the SEC as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

AFEUMR is cleared when the AFEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If AFEUMR is modified during processing, a context error is generated.

14.5.3.2 Host-Provided Context through Prevent Permute

In the default mode of operation, the host provides the key and key size to the AFEU. The initial memory values in the S-Box are permuted with the key to create new S-Box values, which are used to encrypt the plaintext.

If AFEUMR[PP] is set (prevent permute mode is enabled), the AFEU will not require a key. Rather, the host will write the context to the AFEU, and message processing will occur using the provided context. This mode is used to resume processing of a message using the already permuted S-Box. The context may be written through the FIFO if AFEUMR[CS] is set.

14.5.3.2.1 Dump Context

This mode may be independently specified (using AFEUMR[DC]) in addition to host-provided context mode. In this mode, once message processing is complete and the output data is read, the AFEU will make the current context data available for reads through the output FIFO.

NOTE

After the initial key permute to generate a context for an AFEU encrypted session, all subsequent messages will re-use that context, such that it is loaded, modified during the encryption, and unloaded, similar to the use of a CBC initialization vector in DES operations. A new context is generated (through key permute) according to a re-keying interval specified by the security protocol. Context should never be loaded to encrypt a message if a key is loaded and permuted at the same time.

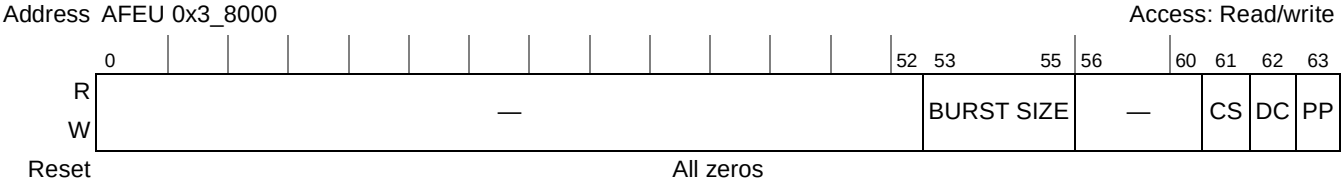


Figure 14-25. AFEU Mode Register (AFEUMR)

Table 14-21 describes AFEU mode register fields.

Table 14-21. AFEUMR Field Descriptions

Bits	Name	Description
0–52	—	Reserved
53–55	BURST SIZE	The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The AFEU signals to the channel that a ‘burst size’ amount of data is available to be pushed to or pulled from the FIFO. **Note:** The inclusion of this field in the AFEU Mode Register is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the AFEU.
56–60	—	Reserved
61	CS	Context source. If set, this causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the context registers. Context source is checked only if the prevent permute (PP) bit is set. 0 Context not from FIFO 1 Context from input FIFO
62	DC	Dump context. If set, this causes the context to be moved from the S-box to the output FIFO following assertion AFEU’s DONE interrupt. 0 Do not dump context 1 After cipher, dump context
63	PP	Prevent permute. Normally, AFEU receives a key and uses that information to randomize the S-box. If reusing a context from a previous descriptor, PP should be set to prevent AFEU from repeating this permutation step. 0 Perform S-box permutation 1 Do not permute

14.5.3.3 AFEU Key Size Register (AFEUKSR)

As shown in Figure 14-26, AFEUKSR indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in AFEUKSR is ignored. AFEUKSR is cleared when the AFEU is reset or re-initialized. If the key size specified is less than 1 or greater than 16, a key size error will be generated. If AFEUKSR is modified during processing, a context error will be generated. Note that although the AFEU supports key lengths as short as 1 byte, a 1-byte key offers little security. Most uses of ARC4 specify keys of 5–16 bytes.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug.

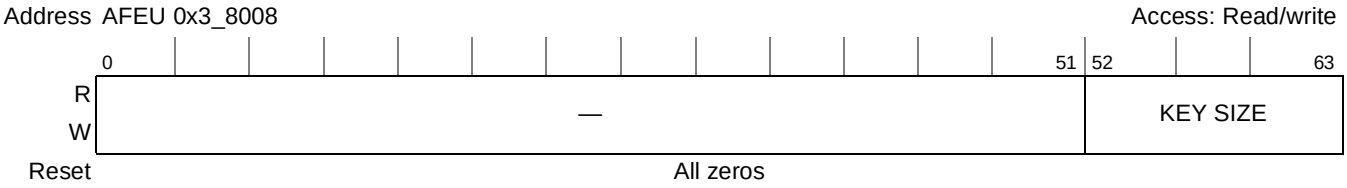


Figure 14-26. AFEU Key Size Register (AFEUKSR)

NOTE

The device driver will create properly formatted descriptors for situations requiring a key permute prior to ciphering. When operating the SEC as a slave (typically while in debug mode), the user must set the AFEU mode register (AFEUMR) to perform ‘permute with key’, then write the key data to the AFEU key registers, then write the key size to the key size register (AFEUKSR[Key Size]). The AFEU will start permuting the memory with the contents of the AFEU key registers immediately after AFEUKSR[Key Size] is written.

14.5.3.4 AFEU Context/Data Size Register (AFEUDSR)

The AFEU context/data size register, shown in Figure 14-27, stores the number of bits in the final message block. AFEUDSR is cleared when the AFEU is reset or re-initialized. The last message block can be between 8 to 64 bits. If a data size that is not a multiple of 8 bits is written, a data size error will be generated. A data size of 0 is illegal and results in the associated crypto-channel locking, requiring a crypto-channel and AFEU reset.

AFEUDSR is also used to specify the context size. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

NOTE

In slave mode, when reloading an existing context, the user must write the context to the input FIFO, then write the context size (always 2072 bits). The write of the context size indicates to the that all context has been loaded. The user then writes the message data size to AFEUDSR. After this write, the user may begin writing message data to the FIFO.

Writing to AFEUDSR signals the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error will be generated.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

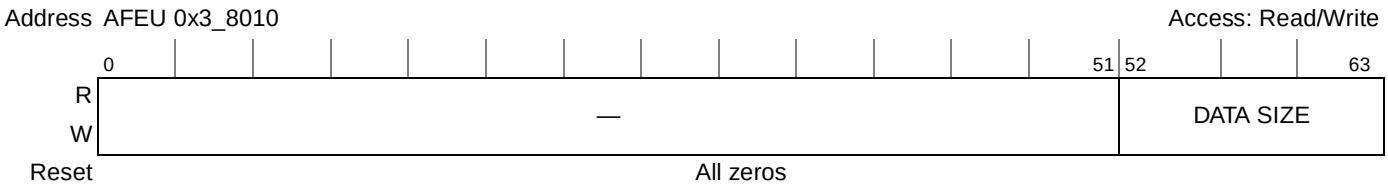


Figure 14-27. AFEU Data Size Register

14.5.3.5 AFEU Reset Control Register (AFEURCR)

The AFEU reset control register, shown in Figure 14-28, allows 3 levels reset that affect the AFEU only, as defined by 3 self-clearing bits. It should be noted that the AFEU executes an internal reset sequence for hardware reset, software reset, or module initialization, which performs proper initialization of the S-Box. To determine when this is complete, observe the RESET_DONE bit in AFEUSR.

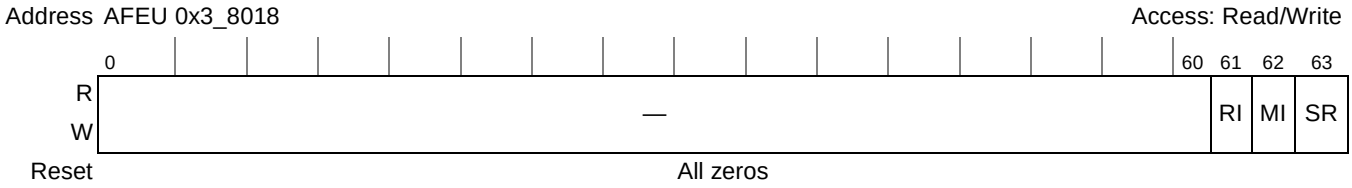


Figure 14-28. AFEU Reset Control Register (AFEURCR)

Table 14-22 describes AFEURCR fields

Table 14-22. AFEURCR Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit causes AFEU interrupts signaling DONE and ERROR to be reset. It further resets the state of AFEUISR. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 Do not reset 1 Reset most of AFEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AFEU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the AFEU enters a routine to perform proper initialization of the S-box. 0 Do not reset 1 Full AFEU reset **Note:** Following a power on reset, the AFEU must be reset prior to first use. Failure to do so results in undefined behavior. Set the SR bit to perform AFEU reset.

14.5.3.6 AFEU Status Register (AFEUSR)

This status register, shown in Figure 14-29, contains 6 bits which reflect the state of the AFEU internal signals.

The AFEUSR is read-only. Writing to this location causes address error being reflected in AFEUISR.

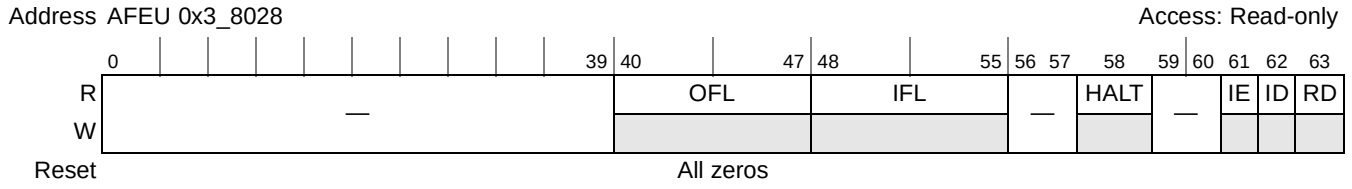


Figure 14-29. AFEU Status Register

Table 14-23 describes AFEUSR fields

Table 14-23. AFEUSR 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 whether the AFEU has halted due to an error. 0 AFEU not halted 1 AFEU halted **Note:** Because the error causing the AFEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	IE	Interrupt error. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling error 1 AFEU is signaling error
62	ID	Interrupt done. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling done 1 AFEU is signaling done
63	RD	Reset done. 0 Reset in progress 1 Reset done. AFEU has completed its internal reset sequence.

14.5.3.7 AFEU Interrupt Status Register (AFEUISR)

The interrupt status register, seen in Figure 14-30, tracks the state of possible errors, if those errors are not masked, via the AFEU interrupt control register.

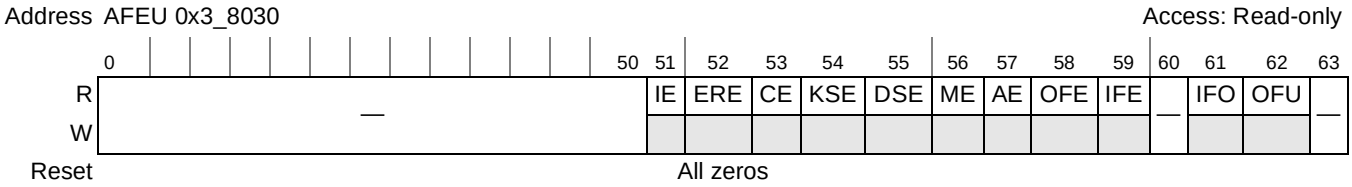


Figure 14-30. AFEU Interrupt Status Register (AFEUISR)

Table 14-24 describes AFEUISR fields.

Table 14-24. AFEUISR Field Descriptions

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error
52	ERE	Early read error. The AFEU Context Memory or Control was read while the AFEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context error. AFEUMR, AFEUKR*n*, AFEUKSR, AFEUDSR, or context memory was modified while AFEU processes data. 0 No error detected 1 Context error
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to AFEUKSR. 0 No error detected 1 Key size error
55	DSE	Data size error. An inconsistent value (not a multiple of 8 bits, or larger than 64 bits) was written to AFEUDSR. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in AFEUMR. Note: Writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEUDSR. 0 No Output FIFO error detected 1 Output FIFO error detected
59	IFE	Input FIFO error. The AFEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No input FIFO error detected 1 Input FIFO error detected
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed. **Note:** When operating as a master, the AFEU implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the AFEU cannot accept FIFO inputs larger than 512 bytes without overflowing.
62	OFU	Output FIFO underflow. The AFEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error.
63	—	Reserved

14.5.3.8 AFEU Interrupt Control Register (AFEUICR)

The interrupt control register, shown in Figure 14-31, controls the result of detected errors. For a given error (as defined in Section 14.5.3.7, "AFEU Interrupt Status Register (AFEUISR)"), if the corresponding AFEUICR bit is set, the error is disabled; no error interrupt occurs and AFEUISR is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, AFEUISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

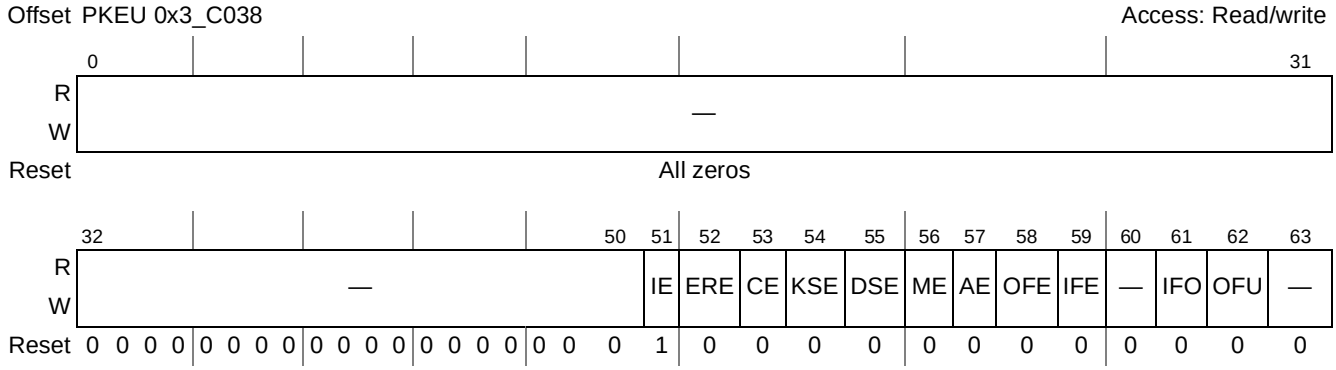


Figure 14-31. AFEU Interrupt Control Register (AFEUICR)

Table 14-25 describes AFEUICR fields.

Table 14-25. AFEUICR Field Descriptions

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The AFEU register was read while the AFEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. AFEUKR*n*, AFEUKSR, AFEUDSR, AFEUMR, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds 1–16 bytes was written to AFEUKSR. 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value was written to the AFEUDSR. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value was detected in AFEUMR. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled

Table 14-25. AFEUICR Field Descriptions (continued)

Bits	Names	Description
58	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEUDSR. 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The AFEU Input FIFO was detected non-empty upon generation of DONE interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The AFEU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The AFEU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.3.9 AFEU End of Message Register (AFEUEMR)

AFEUEMR, shown in Figure 14-32, is used to indicate an ARC4 operation may be completed. After the final message block is written to the input FIFO, AFEUEMR must be written. The value in AFEUDSR will be used to determine how many bits of the final message block (8–64, in multiples of 8) will be processed. Writing to this register causes the AFEU to process the final block of a message, allowing it to signal DONE. If AFEUMR[DC] is set (dump context mode is enabled), the context is written to the output FIFO following the last message word. A read of AFEUEMR always returns a zero value.

AFEUEMR is only used when the AFEU is operated as a slave. The descriptors and crypto-channel activate the AFEU (by means of an internally generated write to AFEUEMR) when the SEC acts as an initiator.

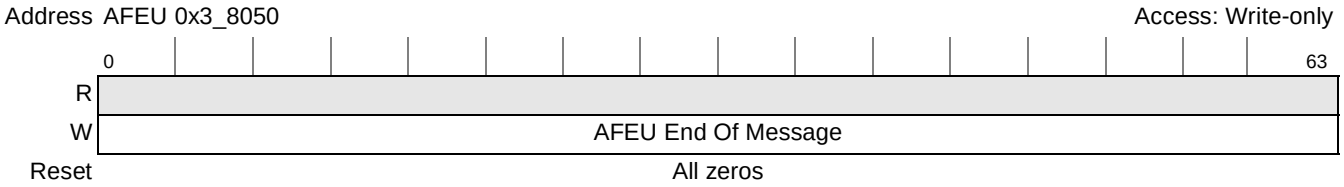


Figure 14-32. AFEU End of Message Register (AFEUEMR)

14.5.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

14.5.3.10.1 AFEU Context Memory

The S-Box memory consists of 32 64-bit words, each readable and writable. The S-Box contents should not be written with data unless it was previously read from the S-Box. Context data may only be written

if AFEUMR[PP] is set (prevent permutation mode is enabled, see Figure 14-25), and the context data must be written prior to the message data. If the context registers are written during message processing or AFEUMR[PP] is not set, a context error will be generated. Reading this memory while the module is not done will generate an error interrupt.

14.5.3.10.2 AFEU Context Memory Pointer Register

The context memory pointer register holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC4 algorithm. If this register is written during message processing, a context error will be generated.

When performing ARC4 operations, the user has the option of performing a new S-Box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing the next packet. The S-Box contents (256 bytes) plus the 3 bytes of the context memory pointers are unloaded and reloaded through the AFEU FIFOs.

AFEU context consists of the contents of the S-Box, as well as three counter values, which indicate the next values to be used from the S-Box. Context must be loaded in the same order in which it was unloaded.

14.5.3.11 AFEU Key Registers (AFEUK0, AFEUK1)

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-Box, in conjunction with the AFEU key size register. AFEU performs permutation starting with the first byte of AFEUK0, and uses as many bytes from two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

14.5.3.11.1 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the AFEU FIFO address space causes the 64-bit-word to be pushed onto the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space causes a 64-bit-word to be popped off the AFEU output FIFO. Overflows and underflows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

14.5.4 Message Digest Execution Unit (MDEU)

The registers used in the MDEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.4.1 MDEU Mode Register (MDEUMR)

The MDEU mode register (MDEUMR) is used to program the function of the MDEU. Bits 56–63 of the MDEUMR are specified by the user through the MODE0 or MODE1 field of the descriptor header. The remaining bits are supplied by the channel and thus are not under direct user control.

The MDEUMR has two configurations, determined by the value of the NEW bit (see Figure 14-33 and Figure 14-34). The ‘old’ configuration (NEW = 0) is used by descriptor types numbered 0000_0, 0001_0, 0010_0, and 1100_0, and is backward compatible with previous versions of SEC 2.0. The ‘new’ configuration (NEW = 1) is used by all other descriptor types.

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.

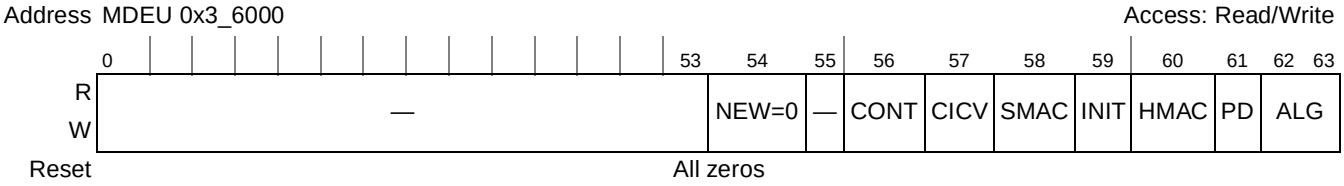


Figure 14-33. MDEU Mode Register (MDEUMR) in ‘Old’ Configuration

Table 14-26 describes MDEUMR fields in ‘old’ configuration.

Table 14-26. 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.

Table 14-26. MDEUMR in 'Old' Configuration (continued)

Bits	Name	Description
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.
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

Address MDEU 0x3_6000 Access: Read/Write

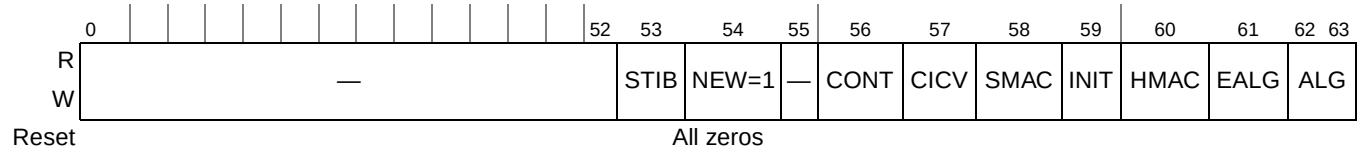


Figure 14-34. MDEU Mode Register (MDEUMR) in 'New' Configuration

Table 14-27 describes MDEUMR fields in 'new' configuration.

Table 14-27. 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.		

Table 14-27. MDEUMR in 'New' Configuration (continued)

Bits	Name	Description
56	CONT	Continue. Most operations will require this bit to be cleared. Set only when the data to be hashed is spread across multiple descriptors. 0 Do autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash will be continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare integrity check values 0 Normal operation; no ICV comparison. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV size register.
58	SMAC	Specifies whether to perform an SSL-MAC operation 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization bit. Most operations will require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit is set on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	EALG	The EALG (Extended Algorithm bit) and ALG (Algorithm) bits together specify the message digest algorithm, as follows: 000 SHA-160 algorithm (full name for SHA-1) 001 SHA-256 algorithm 010 MD5 algorithm 011 SHA-224 algorithm Others: Reserved
62–63	ALG	

14.5.4.2 Recommended Settings for MDEUMR

The most common task likely to be executed through the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPSec, and TLS. The SSL 3.0 protocol uses a slightly different 'SSL-MAC'. If an HMAC or SSL-MAC is to be performed using a single descriptor (with the MDEU acting as sole or secondary EU), the following mode register bit settings should be used:

Table 14-28. Mode Register—HMAC or SSL-MAC Generated by Single Descriptor

Bits	Field	Value	
		for HMAC	for SSL-MAC
56	CONT	0 (off)	0 (off)
58	SMAC	0(on)	1(on)
59	INIT	1(on)	1(on)
60	HMAC	1(on)	0(on)

To generate an HMAC for a message that is spread across a sequence of descriptors, the following mode register bit settings should be used:

Table 14-29. Mode Register—HMAC Generated across a Sequence of Descriptors

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
56	CONT	1 (on)	1 (on)	0 (off)
59	INIT	1 (on)	0 (off)	0 (off)
60	HMAC	1 (on)	0 (off)	1 (on)

All descriptors other than the final descriptor must output the intermediate message digest for the next descriptor to reload as MDEU context.

SSL-MAC operations cannot be spread across a sequence of descriptors.

Additional information on descriptors can be found in Section 14.2.1, “Data Packet Descriptors.”

14.5.4.3 MDEU Key Size Register (MDEUKSR)

MDEUKSR, shown in Figure 14-35, indicates the number of bytes of key memory that should be used in HMAC generation. The MDEU supports at most 64 bytes of key. The MDEU will generate a key size error if the value written to MDEUKSR exceeds 64 bytes.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

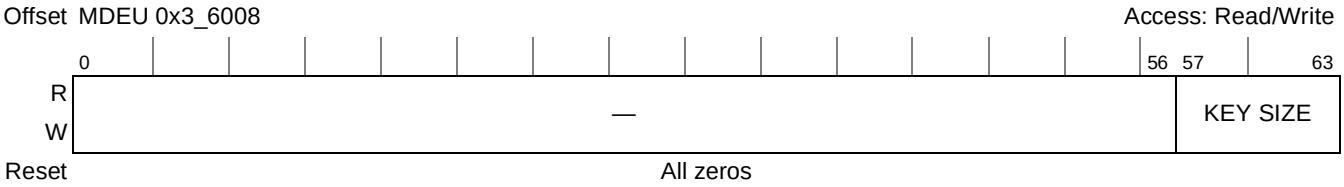


Figure 14-35. MDEU Key Size Register (MDEUKSR)

14.5.4.4 MDEU Data Size Register (MDEUDSR)

The MDEU data size register, shown in Figure 14-36, stores the size of the last block of data (in bits) to be processed. Since the MDEU does not support bit offsets, any value other than 0 in bits 61–63 will cause a data size error. Bits 58–60 are used to identify the ending byte location in the last 8-byte dword. This is used to add the data padding when auto padding is selected. MDEUDSR is cleared when the MDEU is reset, re-initialized, and at the end of processing the complete message.

NOTE

Writing to MDEUDSR allows the MDEU to enter auto-start mode. Therefore, the required context data should be written prior to writing the data size.

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated interrupt control register before performing debug operations.

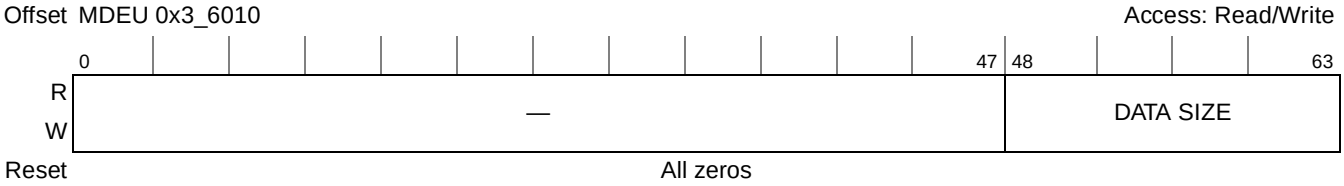


Figure 14-36. MDEU Data Size Register (MDEUDSR)

14.5.4.5 MDEU Reset Control Register (MDEURCR)

MDEURCR, shown in Figure 14-37, allows three levels reset of just the MDEU, as defined by the three self-clearing bits.

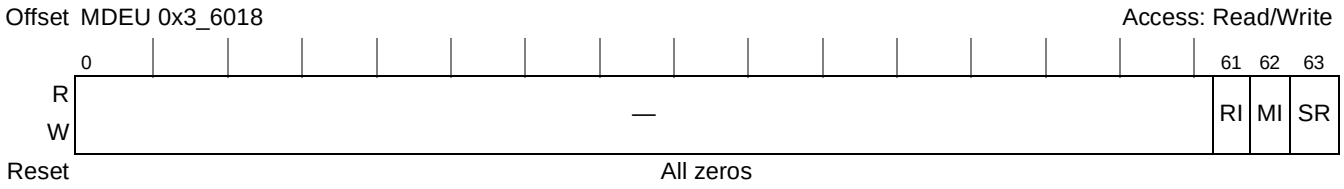


Figure 14-37. MDEU Reset Control Register (MDEURCR)

Table 14-30 describes MDEU reset control register fields.

Table 14-30. MDEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes MDEU interrupts signaling DONE and ERROR to be reset. It further resets the state of the MDEUISR. 0 No reset 1 Reset interrupt logic

Table 14-30. MDEU Reset Control Register Field Descriptions (continued)

Bits	Name	Description
62	MI	Module initialization is nearly the same as software reset, except that MDEUICR is unchanged. 0 No reset 1 Reset most of MDEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset

14.5.4.6 MDEU Status Register (MDEUSR)

MDEUSR reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding and key padding, and are not important to the user; however, the user should be aware that reads of this register, especially during processing, are likely to return non-zero values for many bits between 0–57. The four signals shown are those most likely to be of interest to the user.

MDEUSR, shown in Figure 14-38, is read only.

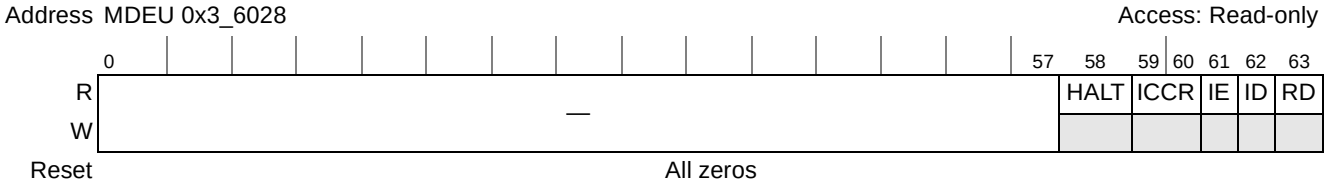


Figure 14-38. MDEU Status Register

Table 14-23 describes MDEUSR fields.

Table 14-31. MDEU Status Register Field Descriptions

Bits	Name	Description
0–57	—	Reserved
58	HALT	Halt. Indicates that the MDEU halted due to an error. 0 MDEU not halted 1 MDEU halted **Note:** Because the error causing the MDEU to stop operating may be masked to MDEUISR, MDEUSR is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved **Note:** A passed or failed result is generated only if ICV checking is enabled.

Table 14-31. MDEU Status Register Field Descriptions (continued)

Bits	Name	Description
61	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller ISR (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 MDEU is not signaling error. 1 MDEU is signaling error.
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller ISR (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 MDEU is not signaling done 1 MDEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that MDEU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

14.5.4.7 MDEU Interrupt Status Register (MDEUISR)

The interrupt status register tracks the state of possible errors, if those errors are not masked through the MDEUISR. The definition of each field in MDEUISR is shown in Figure 14-39.

Address MDEU 0x3_6030 Access: Read-only

	0–48	49	50	51	52	53	54	55	56	57	58–60	61	62–63
R	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
W													
Reset		All zeros											

Figure 14-39. MDEU Interrupt Status Register (MDEUISR)

Table 14-32 describes MDEUISR fields.

Table 14-32. MDEUISR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the MDEU.
50	—	Reserved
51	IE	Internal error. Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected **Note:** This bit is set any time an enabled error condition occurs and can only be cleared by resetting the MDEU.
52	ERE	Early read error. The MDEU context was read before the MDEU completed the hashing operation. 0 No error detected 1 Early read error

Table 14-32. MDEUISR Field Descriptions (continued)

Bits	Name	Description
53	CE	Context error. The MDEU key register, MDEUKSR, or MDEUDSR was modified while MDEU was hashing. 0 No error detected 1 Context error
54	KSE	Key size error. A value greater than 64 bytes was written to MDEUKSR. 0 No error detected 1 Key size error
55	DSE	Data size error. A value not a multiple of 512 bits while the MDEU mode register autopad bit is cleared. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value for ALG was detected in MDEUMR. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 No error detected 1 Address error
58–60	—	Reserved
61	IFO	Input FIFO overflow. The MDEU input FIFO has been pushed while full. 0 No overflow detected 1 Input FIFO has overflowed **Note:** When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 Bytes without overflowing.
62–63	—	Reserved

14.5.4.8 MDEU Interrupt Control Register (MDEUICR)

MDEUICR, shown in Figure 14-40, controls the result of detected errors. For a given error (as defined in Section 14.5.4.7, “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 interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Address MDEU 0x3_6038 Access: Read/Write

	0–48	49	50	51	52	53	54	55	56	57	58–60	61	62	63
R W	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—	
Reset	All zeros	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Figure 14-40. MDEU Interrupt Control Register (MDEUICR)

Table 14-32 describes MDEUICR fields.

Table 14-33. MDEUICR Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the MDEU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error is detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early read error. The MDEU register is read while the MDEU is performing hashing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. The MDEU key register, MDEUKSR, MDEUDSR, or MDEUMR is modified while the MDEU is performing hashing. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. A value outside the bounds of 64 bytes is written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. An inconsistent value is written to MDEUDSR: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error. An illegal value is detected in MDEUMR. 0 Mode error enabled 1 Mode error disabled
57	AE	Address error. An illegal read or write address is detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–60	—	Reserved
61	IFO	Input FIFO overflow. The MDEU input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62–63	—	Reserved

14.5.4.9 MDEU ICV Size Register (MDEUICVSR)

The MDEU ICV size register, shown in Figure 14-41, stores the number of bytes of the ICV result to be compared if the MDEU performs ICV comparison. (See Section 14.5.4.1, “MDEU Mode Register (MDEUMR).”)

This register is cleared when the MDEU is reset or re-initialized.

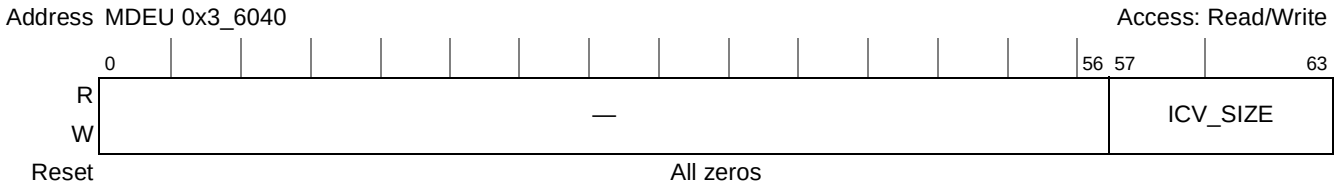


Figure 14-41. MDEU ICV Size Register

14.5.4.10 MDEU EU-Go Register (MDEUEUG)

The EU-Go register in the MDEU, see Figure 14-42, is used to indicate an authentication operation may be completed. After the final message block is written to the input FIFO, the EU-Go register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 512) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.

The DEU EU-Go register is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the MDEU (via an internally generated write to the EU-Go register) when the SEC acts as an initiator.

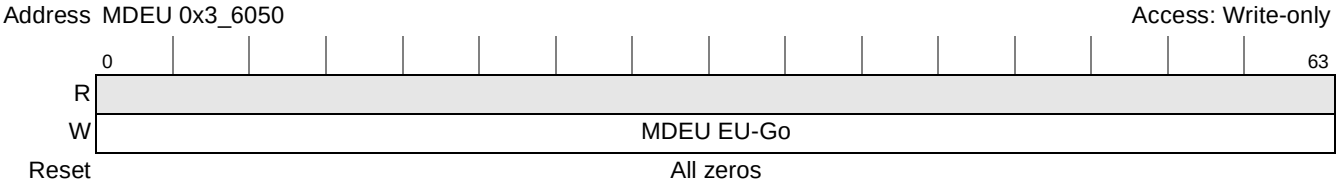


Figure 14-42. MDEU EU-Go Register

14.5.4.11 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bit count.

NOTE

SHA-1and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done will generate an error interrupt.

After a power-on reset, all the MDEU context register values are cleared. Figure 14-43 shows how the MDEU context registers are initialized if the INIT bit is set in the MDEU mode register. All registers are

initialized, regardless of mode selected, however only the appropriate context register values are used in hash generation per the mode selected. The user typically doesn't care about the MDEU Context Register initialization values, however they are documented for completeness in the event the user reads these registers during a debug operation. MDEU reset via the MDEU reset control register (Figure 14-37) or SEC global software reset (Figure 14-72) does not clear these registers.

	0 — 31	32 — 63	
Name	A	B	Context offset 0x3_6100
MD-5	0x01234567	0x89ABCDEF	
SHA-1	0x67452301	0xEFCDAB89	
SHA-256	0x6A09E667	0xBB67AE85	
Name	C	D	Context offset 0x3_6108
MD-5	0xFEDCBA98	0x76543210	
SHA-1	0x98BADCFE	0x10325476	
SHA-256	0x3C6EF372	0xA54FF53A	
Name	E	F	Context offset 0x3_6110
MD-5	0xF0E1D2C3	0x8C68059B	
SHA-1	0xC3D2E1F0	0x9B05688C	
SHA-256	0x510E527F	0x9B05688C	
Name	G	H	Context offset 0x3_6118
MD-5	0xABD9831F	0x19CDE05B	
SHA-1	0x1F83D9AB	0x5BE0CD19	
SHA-256	0x1F83D9AB	0x5BE0CD19	
Name	Message Length Count		Context offset 0x3_6120
Reset	0		

Figure 14-43. MDEU Context Register

14.5.4.12 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

14.5.4.13 MDEU FIFO

MDEU uses an input FIFO to hold data to be hashed. The input FIFO is multiply addressable, but those multiple addresses point only to the write (push) end of the FIFO. A write to anywhere in the MDEU FIFO address space causes the 64-bit-words to be pushed onto the MDEU input FIFO, and a read from anywhere in the MDEU FIFO address space returns all zeros.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

14.5.5 Random Number Generator (RNG)

This section contains details about the random number generator (RNG), including detailed register map, modes of operation, status and control registers, and FIFOs.

The RNG is an execution unit capable of generating 64-bit random numbers. It is designed to comply with the FIPS-140 standard for randomness and non-determinism. A linear feedback shift register (LSFR) and cellular automata shift register (CASR) are operated in parallel to generate pseudo-random data.

The RNG consists of six major functional blocks:

- Bus interface unit (BIU)
- Linear feedback shift register (LFSR)
- Cellular automata shift register (CASR)
- Clock controller
- Six ring oscillators

The states of the LFSR and CASR are advanced at unknown frequencies determined by the two ring oscillator clocks and the clock control. When a read is performed, the oscillator clocks are halted and a collection of bits from the LFSR and CASR are XORed together to obtain the 64-bit random output.

The registers used in the MDEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.5.1 RNG Mode Register (RNGMR)

RNGMR, shown in Figure 14-44, is used to control the RNG. One operational mode, randomizing, is defined. Writing any other value than 0 to 56:63 results in a data error interrupt reflected in the RNG interrupt status register. The mode register also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

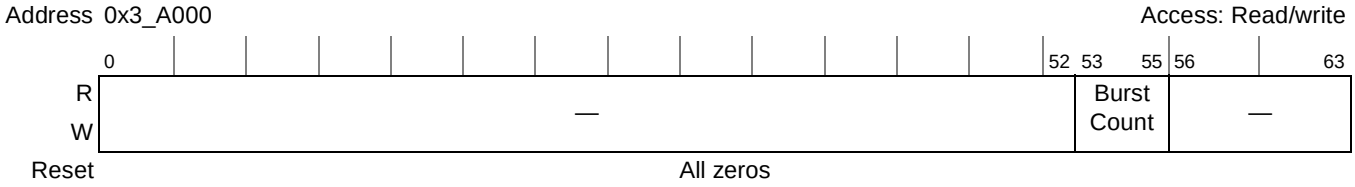


Figure 14-44. RNG Mode Register

Table 14-34. RNG Mode Register Definitions

Bits	Name	Description
0–52	—	Reserved, must be set to zero.
53–55	Burst Count	Burst Count. Implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The RNG signals to the crypto-channel that a ‘Burst Size’ amount of data is available to be pulled from the FIFO. **Note:** The inclusion of this field in the RNG mode register is to avoid confusing a user who may read this register in debug mode. Burst Size should not be written directly to the RNG.
56–63	—	Reserved

14.5.5.2 RNG Data Size Register (RNGDSR)

RNGDSR, shown in Figure 14-45, is used to tell the RNG to begin generating random data. The actual contents of the data size register does not affect the operation of the RNG. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once the data size register is written, the RNG will begin pushing data onto the FIFO. Data will be pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG then attempts to keep the FIFO full.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

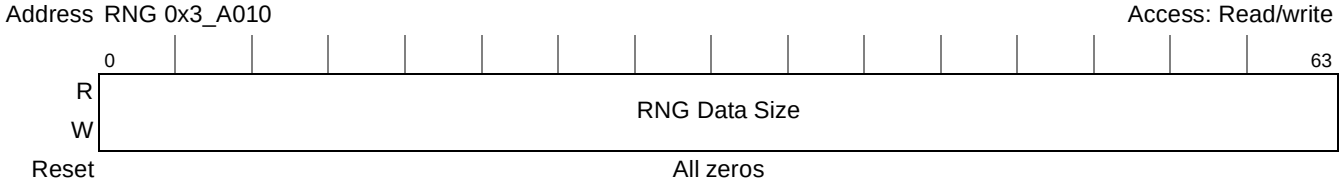


Figure 14-45. RNG Data Size Register

14.5.5.3 RNG Reset Control Register (RNGRCR)

RNGRCR, shown in Figure 14-46, contains three reset options specific to the RNG.

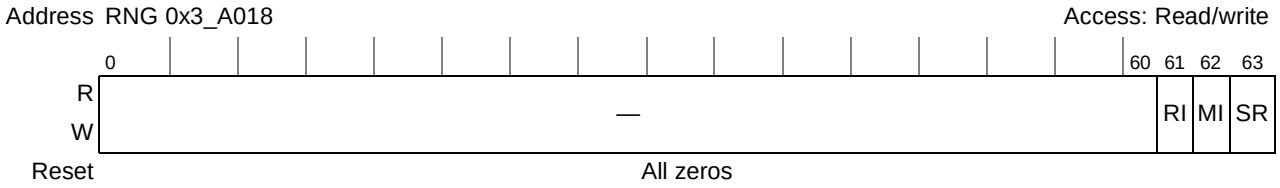


Figure 14-46. RNG Reset Control Register

Table 14-35 describes RNG reset control register fields.

Table 14-35. RNG Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes RNG interrupts signalling DONE and ERROR to be reset. It further resets the state of the RNG interrupt status register. 0 No reset 1 Reset interrupt logic
62	MI	Module Initialization. This reset value performs enough of a reset to prepare the RNG for another request, without forcing the internal control machines and the output FIFO to be reset, thereby invalidating stored random numbers or requiring reinvocation of a warm-up period. Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 No reset 1 Reset most of RNG
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the RNG. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNG reset

14.5.5.4 RNG Status Register (RNGSR)

RNGSR, shown in Figure 14-47, contains 6 fields which reflect the state of the RNG internal signals.

Writing to this location causes an address error being reflected in the RNG interrupt status register.

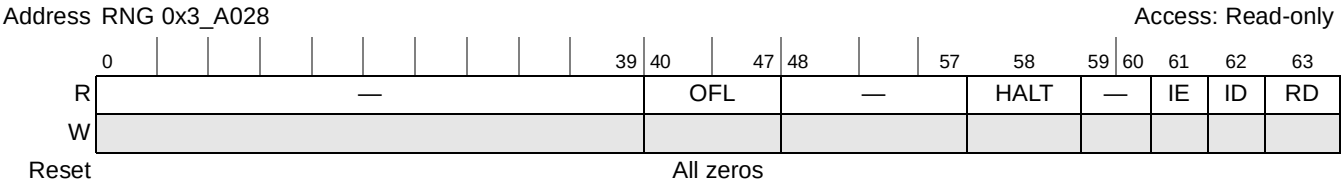


Figure 14-47. RNG Status Register

Table 14-23 describes RNG status register fields.

Table 14-36. RNG Status Register Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–57	—	Reserved. Internal status bits may be observed as non-zero.
58	HALT	Halt. Indicates that the RNG has halted due to an error. 0 RNG not halted 1 RNG halted Note: Because the error causing the RNG to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	IE	Interrupt Error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 RNG is not signaling error 1 RNG is signaling error
62	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 RNG is not signaling done 1 RNG is signaling done
63	RD	Reset Done. This status bit, when high, indicates that the RNG has completed its internal reset sequence. 0 Reset in progress 1 Reset done

14.5.5.5 RNG Interrupt Status Register (RNGISR)

RNGISR, shown in Figure 14-48, tracks the state of possible unmasked errors via RNGICR.

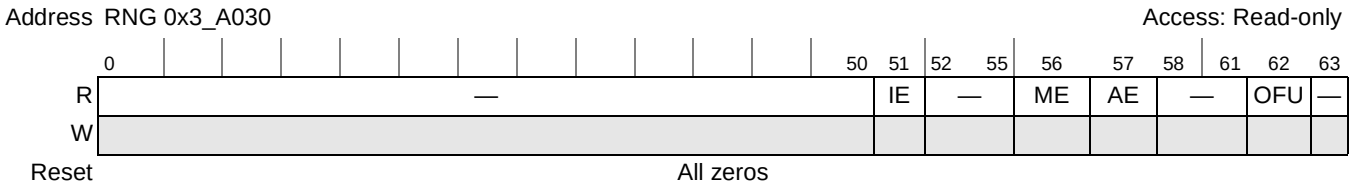


Figure 14-48. RNG Interrupt Status Register

Table 14-37 describes RNGISR fields.

Table 14-37. RNG Interrupt Status Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal Error 0 No internal error detected 1 Internal error

Table 14-37. RNG Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
52–55	—	Reserved
56	ME	Mode Error. Indicates that the host has attempted to write an illegal value to the mode register 0 = Valid data 1 = Invalid data error
57	AE	Address Error. An illegal read or write address was detected within the RNG address space. 0 No error detected 1 Address error
58–61	—	Reserved
62	OFU	Output FIFO Underflow. The RNG Output FIFO has been read while empty. 0 No overflow detected 1 Output FIFO has underflowed
63	—	Reserved

14.5.5.6 RNG Interrupt Control Register (RNGICR)

RNGICR controls the result of detected errors. For a given error (as defined in Section 14.5.5.5, “RNG Interrupt Status Register (RNGISR)”), if the corresponding RNGICR bit is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, RNGISR is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

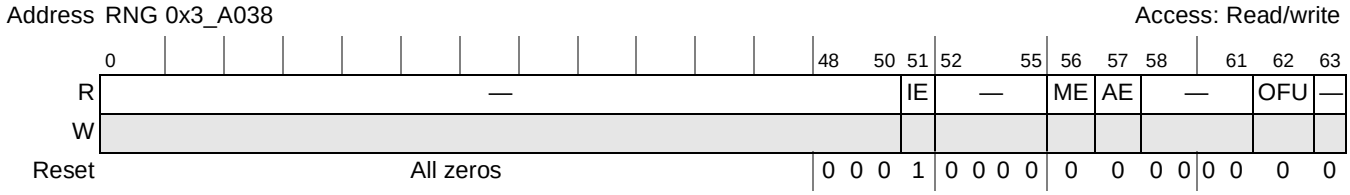


Figure 14-49. RNG Interrupt Control Register

Table 14-38 describes RNG interrupt status register fields.

Table 14-38. RNG Interrupt Control Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while generating random numbers. 0 Internal error enabled 1 Internal error disabled
52–55	—	Reserved
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled

Table 14-38. RNG Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
57	AE	Address Error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–61	—	Reserved
62	OFU	Output FIFO Underflow. RNG Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.5.7 RNG EU-Go Register (RNGEUG)

RNGEUG, shown in Figure 14-50, is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

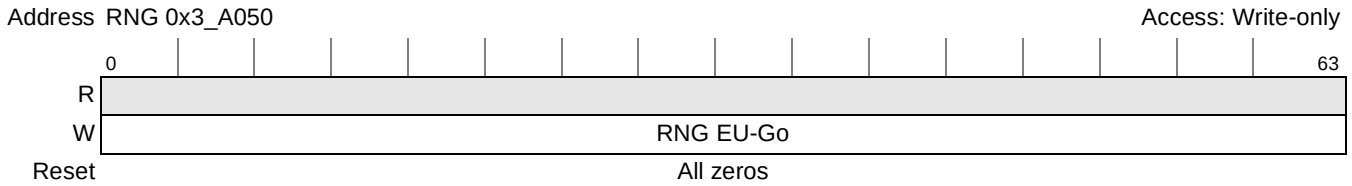


Figure 14-50. RNG EU-Go Register

14.5.5.8 RNG FIFO

RNG uses an output FIFO to collect periodically sampled random 64-bit-words, with the intent that random data always be available for reading. The FIFO is multiply addressed, but those multiple addresses point only to the appropriate end of the output FIFO. A read from anywhere in the RNG FIFO address space causes a 64-bit-word to be popped off of the RNG output FIFO. Underflows caused by reading or writing the RNG output FIFO are reflected in the RNG interrupt status register. Also, a write to the RNG output FIFO space will be reflected as an addressing error in the RNG interrupt status register.

14.5.6 Advanced Encryption Standard Execution Units (AESU)

This section contains details about the Advanced Encryption Standard Execution Units (AESU), including detailed register map, modes of operation, status and control registers, and FIFOs. The registers used in the AESU are documented primarily for debug and slave mode operations. If the SEC requires the use of the AESU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

14.5.6.1 AESU Mode Register (AESUMR)

The AESU mode register, shown in Figure 14-51, contains 7 bits which are used to program the AESU. The mode register is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

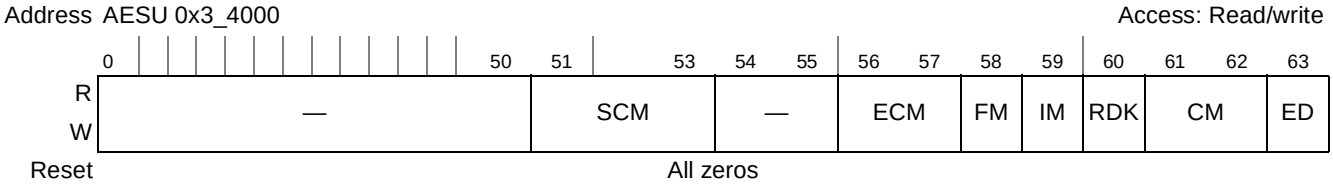


Figure 14-51. AESU Mode Register

Table 14-39 describes AESU mode register fields.

Table 14-39. AESU Mode Register 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 14-40 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 14-40 for mode bit combinations.
63	ED	Encrypt/Decrypt. If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption Note: This bit is ignored if CM is set to “11” - CTR Mode.

SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0’srtp’. See Section Figure 14-59., “AESU Context Register,” for more information on how SRT mode reduces context loading overhead.

Table 14-40. AES Cipher Modes

Mode	ECM (56:57)	CM (61:62)
ECB	00	00
CBC	00	01
Res	xx	10
CTR	00	11
SRT[1]	01	11
CCM (without ICV comparison)	10	00
CCM (with ICV comparison)	11	00
XOR	11	11
All Others	Reserved	

[1] SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0'srtp'. See Section 14.5.6.9.3, "Context for SRT Mode," for more information on how SRT mode reduces context loading overhead.

NOTE

Restore decrypt key (RDK)—In most networking applications, the decryption of an AES protected packet will be performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.)

The use of RDK is completely optional, as the Input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is "0100_0- AESU Key Expand Output". The AESU mode must be "Decrypt". See Table 14-6. for more information. The descriptor will cause the SEC to write the contents of the Context registers and the key registers (containing the expanded decrypt key) to memory.

To process the remainder of the message, use a 'common' descriptor type (0001_0), and set the restore-decrypt-key mode bit. Load the context registers and the expanded decrypt key with previously saved key and context data from the first message. The key size is written as before (16, 24, or 32 bytes).

14.5.6.2 AESU Key Size Register (AESUKSR)

The AESU key size register stores the number of bytes in the key (16,24,32). Any key data beyond the number of bytes in the key size register will be ignored. 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.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

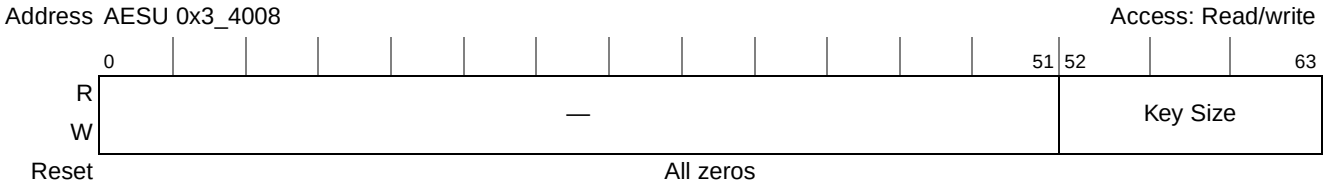


Figure 14-52. AESU Key Size Register

14.5.6.3 AESU Data Size Register (AESUDSR)

AESUDSR is used to advise the AESU of the size of the data to be processed. Depending on the AES mode selected, data size must be divisible by a specific block size or a data size error will occur. In ECB, CBC, and CTR mode, the AESU does not automatically pad messages out to 128-bit blocks, therefore when operating in ECB, CBC, or CTR mode, the message processed by the AESU must be divisible by 128-bits or a data size error will occur. When operating in CCM mode, data size must be divisible by 8-bits or a data size error will occur. In XOR mode the data size must be a multiple of 256 bits (32 bytes). If an improper data size is written, a data size error is generated. Only the lowest 3, 7, or 8 bits of the data size register are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

In normal operation, the full message length to be encrypted or decrypted with the AESU is copied from the descriptor to the AESU data size register, however only bits 56:63 are checked to determine if there is a data size error. If 57:63 are all zeroes, the message is evenly divisible into 128-bit blocks.

This register is cleared when the AESU is reset or re-initialized. If a data size other than 128-bits is specified, an illegal data size error will be generated. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error will be generated.

NOTE

Writing to this register while in debug mode generates an illegal size error. Disable the illegal size error in the associated Interrupt Control Register prior to performing debug operations.

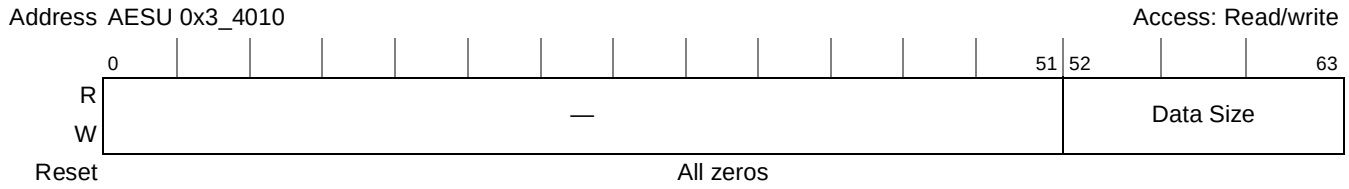


Figure 14-53. AESU Data Size Register

14.5.6.4 AESU Reset Control Register (AESURCR)

AESURCR allows three levels reset of just AESU, as defined by the three self-clearing bits:

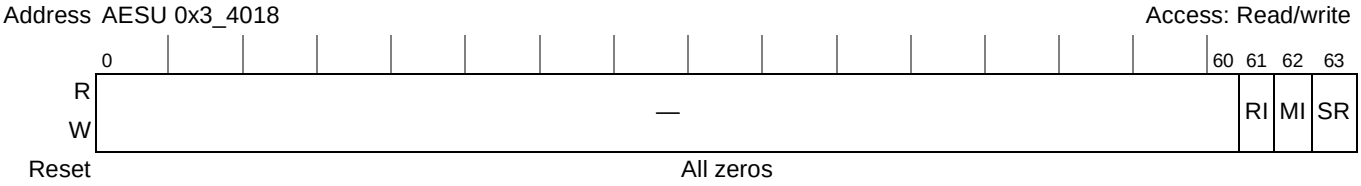


Figure 14-54. AESU Reset Control Register

Table 14-17 describes AESU reset control register fields.

Table 14-41. AESU Reset Control Register Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes AESU interrupts signalling DONE and ERROR to be reset. It further resets the state of the AESU interrupt status register. 0 Don't reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register 0 Don't reset 1 Reset most of AESU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register will indicate when this initialization routine is complete 0 Don't reset 1 Full AESU reset

14.5.6.5 AESU Status Register (AESUSR)

AESUSR, shown in Figure 14-55, reflects the state of six status outputs. Writing to this location causes an address error being reflected in the AESU interrupt status register.

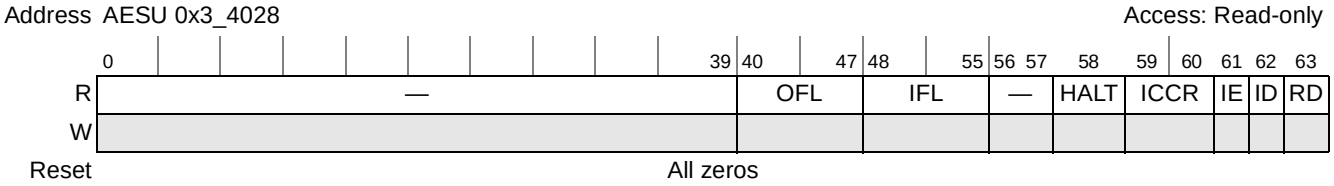


Figure 14-55. AESU Status Register

Table 14-23 describes AESU status register fields.

Table 14-42. AESU Status Register 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 AESU has halted due to an error. 0 AESU not halted 1 AESU halted **Note:** Because the error causing the AESU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved A passed or failed result is generated only if ICV checking is enabled and the cipher mode selected is CCM with ICV comparison
61	IE	Interrupt Error.This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 AESU is not signaling error 1 AESU is signaling error
62	ID	Interrupt Done. This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 14.7.2.2, “Interrupt Status Register (ISR)”). 0 AESU is not signaling done 1 AESU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that AESU has completed its internal reset sequence. 0 Reset in progress 1 Reset done

14.5.6.6 AESU Interrupt Status Register (AESUISR)

The AESU interrupt status register tracks the state of possible errors, if those errors are not masked, via the AESU interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 14-56.

Address AESU 0x3_4030 Access: Read-only

	0–48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
W																
Reset	All zeros															

Figure 14-56. AESU Interrupt Status Register

Table 14-19 describes AESU interrupt register fields.

Table 14-43. AESU Interrupt Status Register 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 AESU.
50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while the AESU was processing. 0 No error detected 1 Internal error **Note:** This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the AESU.
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 32bytes) was written to the AESU Key Size Register 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 multiple of 128 bits. 0 No error detected 1 Data size error
56	ME	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Valid Data 1 Reserved or invalid mode selected
57	AE	Address Error. An illegal read or write address was detected within the AESU address space. 0 No error detected 1 Address error

Table 14-43. AESU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
58	OFE	Output FIFO Error. The AESU output FIFO was detected non-empty upon write of AESU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO Error. The AESU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed **Note:** When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the cannot accept FIFO inputs larger than 512 Bytes without overflowing.
62	OFU	Output FIFO Underflow. The AESU Output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

14.5.6.7 AESU Interrupt Control Register (AESUICR)

The AESU interrupt control register, shown in Figure 14-57, controls the result of detected errors. For a given error (as defined in Section 14.5.6.6, “AESU Interrupt Status Register (AESUISR)”), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

Address AESU 0x3_4038 Access: Read/write

	0–‘47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R W	—		ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	All zeros	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14-57. AESU Interrupt Control Register

Table 14-44 describes the AESU interrupt control register fields.

Table 14-44. AESU Interrupt Control Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. The supplied ICV did not match the one computed by the AESU. 0 Integrity check error enabled 1 Integrity check error disabled
50	—	Reserved

Table 14-44. AESU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
51	IE	Internal Error. An internal processing error was detected while the AESU was processing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The AESU IV Register was read while the AESU was processing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while the AESU was processing. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
58	OFE	Output FIFO Error. The AESU Output FIFO was detected non-empty upon write of AESU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO Error. The AESU Input FIFO was detected non-empty upon generation of DONE interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO Underflow The AESU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

14.5.6.8 AESU End of Message Register (AESUEMR)

AESUEMR, shown in Figure 14-58, is used to indicate an AES operation may be completed. After the final message block is written to the input FIFO, the end of message register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 128) will be processed. Writing to this register causes the AESU to process the final block of a message, allowing it

to signal DONE. A read of this register always returns a zero. AESUEMR is used only when the is operated as a slave. The descriptors and crypto-channel activate the AESU (via an internally generated write to the end of message register) when the SEC acts as an initiator.

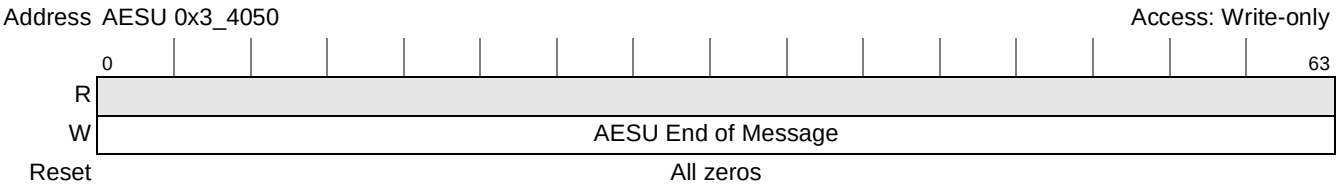


Figure 14-58. AESU End of Message Register

14.5.6.9 AESU Context Registers

There are three 64-bit context data registers that allow the host to read/write the contents of the context used to process the message. The context must be written prior to the key data. If the context registers are written during message processing, a context error will be generated. All context registers are cleared when a hard/soft reset or initialization is performed.

The context registers must be read when changing context and restored to their original values to resume processing an interrupted message (CBC, CTR and CCM modes). For CTR and CCM mode, all seven 64-bit context registers must be read to retrieve context, and all seven must be written back to restore context. Effectively, the user must read the four empty ‘place holder’ context registers in addition to the three context registers holding the Counter and Counter Modulus when in CTR mode. The contents of the ‘empty’ context registers need not be preserved, but when restoring the CTR mode context, the ‘empty’ registers must be filled with 32 bytes of zeros before writing the saved Counter and Counter Modulus.

Context should be loaded with the lower bytes in the lowest 64-bit context register. The context registers are summarized in Figure 14-59.

Context Register (64-bits each)

Cipher Mode	1	2	3	4	5	6	7
ECB	—	—	—	—	—	—	—
CBC	IV1[1]	IV2[1]	—	—	—	—	—
CTR	—	—	—	—	Counter [1]		Counter Modulus[1]
SRT	Counter [1]		Counter Modulus[1]	—	—	—	—
CCM	IV[1] / MAC Tag		Encrypted MAC[2]/Decrypted MAC/Encrypted Counter		Counter [1]		Counter Modulus[1,3]

1 Must be written at the start of a new message

2 Must be written at start of new CCM decryption

3 Header size/MAC size is only used if AES-CCM processing is suspended and resumed.

Figure 14-59. AESU Context Register

14.5.6.9.1 Context for CBC Mode

Within the Context register, for use in CBC mode, are two 64-bit context data registers that allow the host to read/write the contents of the initialization vector (IV):

- IV1 holds the 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 14.5.6.5, "AESU Status Register (AESUSR)." If the IV registers are read prior to assertion of Interrupt Done, an early read error will be generated.

The IV registers must be read when changing context and restored to resume processing an interrupted message (CBC mode only).

14.5.6.9.2 Context for Counter Mode

In counter mode, a random 128-bit initial counter value is incremented modulo 2^n with each block processed. The modulus size can be set between 2^8 through 2^{128}, by powers of 8. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext.

In CTR mode, the block counter is incremented modulo 2^M. The value of M is specified by writing to Context Register 3 as described in Table 14-45.

Value Written	Modulus	Value Written	Modulus
8	2^8	72	2^{72}
16	2^{16}	80	2^{80}
24	2^{24}	88	2^{88}
32	2^{32}	96	2^{96}
40	2^{40}	104	2^{104}
48	2^{48}	112	2^{112}
56	2^{56}	120	2^{120}
64	2^{64}	128	2^{128}

Table 14-45. Counter Modulus

14.5.6.9.3 Context for SRT Mode

As was noted in the AESU mode register, SRT is not a new AES mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0'srtp'. As with counter mode, a random 128-bit initial counter value is incremented modulo 2^n with each block processed. The modulus size can be set between 2^8 through 2^{128}, by powers of 8. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the

ciphertext, or with the ciphertext to recover the plaintext. The block counter is incremented modulo 2^M. The value of M is specified by writing to Context Register 3 as described in Table 14-45.

The only difference between SRT mode and CTR mode is in SRT mode, the AES Context is loaded and read via Context Registers 1–3, with no requirement to access Context Registers 4–7. In CTR mode, Context Registers 1–4 must be loaded with zeros, with the Counter and Modulus being loaded into and read from Context Registers 5–7.

14.5.6.9.4 Context for CCM Mode

The SEC AESU is capable of performing single pass encryption and MAC generation. The host is required to order the CCM context is such a way that the context can be fetched as a contiguous string into the context registers, prior to encryption/MAC generation or decryption/MAC validation. A complete explanation of the context and ordering can be found below.

The context for CCM encryption/MAC generation is:

Reg 1–2 Session specific128 bit Initialization Vector (from memory)

Reg 3–4 128 bits of zero padding

Reg 5–6 Session Specific Counter (Initial Counter Value) (from memory)

Reg 7 Counter Modulus Should be fixed at 0x0000_0080.

Note: The counter modulus for CCM mode is currently defined as 2^128. This value has been made programmable in the SEC in case the final version of 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 symetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symetric 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 symetric key is the MAC Tag. The full 128bits of MAC data is written to context registers 1–2, for use in the next phase of CCM processing. Once the MAC Tag has been generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in Counter mode.
4. The first item to be encrypted in counter mode is the Counter (Initial Counter Value) from Context Registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC Tag (retrieved from Context Reg 1–2) to produce the Encrypted MAC, which is then stored in Context Registers 3–4. At the completion of CCM encrypt processing, this Encrypted MAC is output to memory (per the descriptor pointer) for the host to append to the 802.11i frame. Note: The Encrypted MAC written out to memory by the AESU is the full 128-bits. The host must only append the most significant 64-bits to the frame as the encrypted MAC.

5. The counter value is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of plaintext to produce the first block of cipher text. The ciphertext is placed in the AESU output FIFO.
6. The counter continues to be incremented, and encrypted with the symmetric key, with the result hashed with each successive block of plaintext, until all plaintext has been converted to ciphertext. The SEC controller will manage FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the encrypted MAC has been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is as follows:

Reg 1–2 Session specific128 bit Initialization Vector (from memory)

Reg 3–4 Encrypted MAC (from received frame) + 64 bits of zero padding

Reg 5–6 Session Specific Counter (Initial Counter Value) (from memory)

Reg 7 Counter Modulus Should be fixed at 0x0000_0080.

NOTE

The counter modulus for CCM mode is currently defined as 2^128. This value has been made programmable in the SEC to in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU Context Register prior to CCM decryption.

CCM decryption processing is the reverse of encryption;

With the session specific key and context, the AESU will perform the following operations.

1. Initialize the IV, and encrypt with the symmetric key. Simultaneously, the counter (Initial Counter Value) from Context Registers 5–6 is encrypted with the symmetric key. The result is hashed with the Encrypted MAC (from Context Register 3–4), and the resulting Original MAC is written to Context Reg 3–4, overwriting the Encrypted MAC.
   Note: Strictly speaking, the Counter is encrypted with the symmetric key, however the AESU should be set for ‘decrypt’ to perform the counter and CBC processes in the correct order.
2. The 802.11 frame header is hashed with the encrypted IV. (The AESU automatically determines the header length.) Simultaneously, the counter is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of ciphertext to produce the first block of plaintext. The plaintext is placed in the AESU output FIFO, while simultaneously, in CBC fashion, a copy of the first block of plaintext is hashed with the output of encryption of the 802.11 frame header. The output is encrypted with the symmetric key.
3. As each ciphertext block is converted to plaintext, the plaintext is CBC encrypted. When the final plaintext block has been processed, the CBC MAC (MAC Tag) is written to Context Registers 1–2. The first 64 bits of the MAC Tag are compared to the MAC Tag recovered in step 1.

NOTE

For both encrypt and decrypt operations, if the 802.11 frame is being processed as a whole (not split across multiple descriptors), the ‘Initialize’ and ‘Final MAC’ bits should be set in the AESU Mode Register.

14.5.6.9.5 AESU Key Registers

The AESU key registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to key 1. Any key data written to bytes beyond the value written to the key size register will be ignored. The key data registers are cleared when the AESU is reset or re-initialized. If these registers are modified during message processing, a context error will be generated.

The key data registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the ‘restore decrypt key’ bit must be set in the mode register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

14.5.6.9.6 AESU FIFOs

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted with the key and initialization vector (CBC mode only) and the results are placed in the output FIFO. The output size is the same as the input size.

Writing to the FIFO address space places 64 bits of message data into the input FIFO. The input FIFO may be written any time the IFW signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes of available space is at or above the threshold specified in the mode register. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the data size register.

Reading from the FIFO address space will pop 64 bits of message data from the output FIFO. The output FIFO may be read any time the OFR signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes in the output FIFO is at or above the threshold specified in the mode register.

14.6 Crypto-Channels

A channel in the SEC manages the execution of each cryptographic task, making use of one or more of the SEC’s execution units (EUs). Control information and data pointers for a given task are stored in the form of a descriptor (see Section 14.4.1, “Descriptor Structure”) in system memory or in the channel itself. A descriptor determines what EUs will be used, how they will be configured, where to fetch needed data, and where to store the results. To invoke cryptographic tasks, the host constructs a descriptor, selects a channel, and writes a pointer to the descriptor into the selected channel’s Fetch FIFO. Operations performed by channels include the following (not necessarily in this order):

- If the channel is idle and its fetch FIFO is non-empty, read the next descriptor pointer from the Fetch FIFO, and use this pointer to read the descriptor into the channel’s descriptor buffer.
- Request from the controller the assignment of one or more EUs for the exclusive use of the channel. Where necessary, configure the secondary EU to snoop input or output data intended for the primary EU.

- 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 32Kbytes) from system memory into assigned EU input registers and FIFOs. This may involve using link tables to gather input data that has been split into multiple segments which are stored in various locations of system memory.
- Transfer data parcels (up to 32Kbytes) 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. For the RAID-XOR descriptor type, the channel rotates among three data sources, fetching 32 bytes from each source.
- Initialize the EU-Go register (where applicable) in the assigned EU upon completion of last EU write indicated by the descriptor. The channel will wait for a indication from the EU that processing of input text-data is complete before proceeding with further activity after writing EU-Go.
- Reset assigned EU(s).
- Release assigned EU(s).
- When a descriptor has been completely processed, provide feedback to the host, in the form of interrupt and/or descriptor header write-back to system memory.
- When descriptor processing is halted due to an error, provide feedback to the host via interrupt.

The channel will wait indefinitely for the controller to complete a requested activity before continuing to the next step of descriptor processing.

14.6.1 Crypto-Channel Registers

Crypto-channel registers are described in the following sections.

14.6.1.1 Crypto-Channel Configuration Register (CCCR)

CCCR contains five operational bits permitting configuration of the crypto-channel as shown in Figure 14-60. Table 14-46 describes the CCCR.

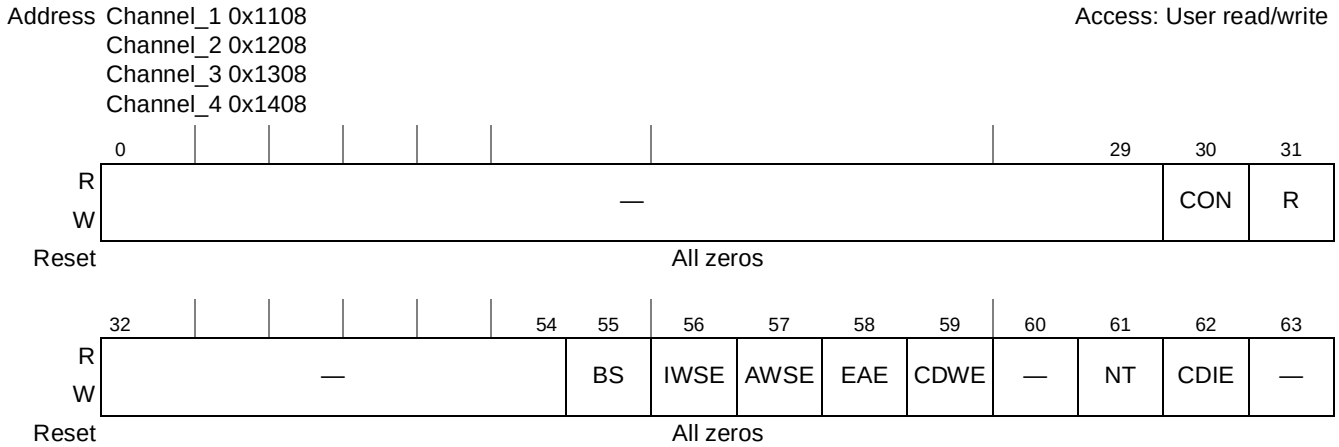


Figure 14-60. Crypto-Channel Configuration Register (CCCR)

Table 14-46. CCCR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, set to zero
30	CON	Continue bit 0 No special action. 1 Causes the same channel reset actions as bit R, except that the fetch FIFO and the lower half of the CCR register are not cleared. After the reset sequence is complete, this bit automatically returns to 0 and the channel resumes normal operation, servicing the next descriptor pointer in the fetch FIFO, if any.
31	R	Reset channel 0 No special action. 1 Causes a software reset of the channel, clearing all its internal state. The details of the software reset actions depend upon what the channel is doing when the bit is set: If the R bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request by asserting the release output signals. The channel then resets all its registers, clears the R bit, and return the channel state machine to the idle state. If the R bit is set after the channel has been assigned an EU, the channel requests a write from the controller to set the software reset bit of the EU. If a secondary EU has been reserved, the channel requests a write to reset that EU as well. The channel next asserts the appropriate release signal to notify the controller that the channel has finished with the reserved EU(s). The channel then resets all the registers, clears the RESET bit and returns the channel state machine to the idle state.
32–54	—	Reserved, set to zero
55	BS	Burst size— The SEC accesses long text-data parcels in main memory through bursts of programmable size: 0 Burst size is 64 bytes 1 Burst size is 128 bytes
56	IWSE	ICV writeback status enable 0 No special action. 1 If the descriptor calls for ICV comparison, then at the completion of descriptor processing, write back the status of all EUs into the header dword.
57	AWSE	Always writeback status enable 0 No special action. 1 At the completion of processing each descriptor, write back the status of all EUs into the header dword. In this case, IWSE has no effect.
59	CDWE	Channel done writeback enable 0 Channel done writeback disabled. 1 Channel done writeback enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, notify the host by writing back the descriptor header with the writeback information shown in Figure 14-61. This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed.
60	—	Reserved, set to zero
61	NT	Notification type. This bit controls when the channel will generate channel done notification. Channel done notification can take the form of an interrupt or modified header writeback or both, depending on the state of the CDIE and CDWE control bits. 0 Global notification. The channel will generate channel done notification (if enabled) at the end of each descriptor. 1 Selected notification. The channel will generate channel done notification (if enabled) at the end of every descriptor with the DONE bit set in the descriptor header.

Table 14-46. CCCR Field Descriptions (continued)

Bits	Name	Description
62	CDIE	Channel done interrupt enable 0 Channel done interrupt disabled 1 Channel Done Interrupt enabled. Upon completion of descriptor processing, if the NT bit is set for Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, then notify the host by asserting an interrupt. Refer to Section 14.6.2, "Interrupts," for complete description of channel interrupt operation.
63	—	Reserved, set to zero

Figure 14-61 shows the format of the header dword when the channel is configured to perform done notification via header writeback. A detailed description of the header dword fields used in writeback notification is provided in Table 14-47.

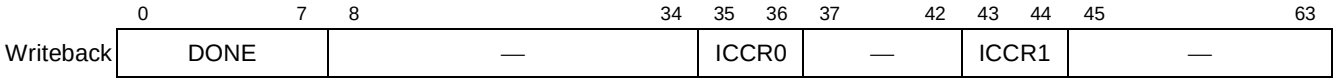


Figure 14-61. Header Dword Writeback Format

Table 14-47. Header Dword Writeback Field Descriptions

Bits	Name	Description
0–7	DONE	When done writeback is used, then at the completion of descriptor processing this byte is written with the value 0xFF. To determine when done writeback is used, see the CDWE, NT, and CDIE fields in the channel configuration register (see Table 14-47).
8–34	—	Reserved.
35–36	ICCR0	Integrity check comparison result from primary. These bits are supplied by the primary EU when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
37–42	—	Reserved.
43–44	ICCR1	Integrity check comparison result from secondary. These bits are supplied by the secondary EU (if any) when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
45–63	—	Reserved.

14.6.1.2 Crypto-Channel Pointer Status Register (CCPSR)

CCPSR contains status fields and counters which provide the user with status information regarding the channel's actual processing of a given descriptor.

Address Channel_1 0x01110
Channel_2 0x01210
Channel_3 0x01310
Channel_4 0x01410

Access: Read-only

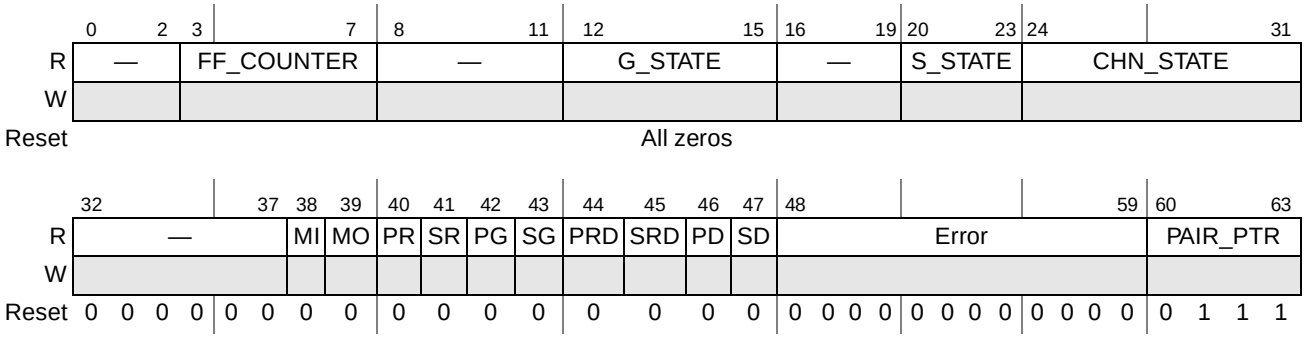


Figure 14-62. Crypto-Channel Pointer Status Register

Table 14-48 describes the crypto-channel pointer status register fields.

Table 14-48. Crypto-Channel Pointer Status Register Signals

Bits	Name	Description
0–2	—	Reserved.
3–7	FF_COUNTER	Fetch FIFO Counter. The fetch FIFO can store up to 24 pointers to descriptors in system memory. This 5 bit counter indicates how many fetch pointers are currently stored in the fifo.
8–11	—	Reserved.
12–15	G_STATE	Gather state machine state. Indicates which stage the crypto-channel is while performing the gather function. Table 14-49 shows the meaning of all possible values of G_STATE. **Note:** G_state is documented for information only. The user typically does not care about the gather state machine.
16–19	—	Reserved.
20–23	S_STATE	Scatter state machine state. Indicates which stage the crypto-channel is while performing the scatter function. Table 14-50 shows the meaning of all possible values of S_STATE. **Note:** S_state is documented for information only. The user typically does not care about the scatter state machine.
24–32	CHN_STATE	Crypto-channel state machine. Indicates the stage of the crypto-channel in the sequence of fetching and processing data descriptors. Table 14-49 shows the meaning of all possible values of STATE. **Note:** CHN_State is documented for information only. The user typically does not care about the crypto-channel state machine.
31–37	—	Reserved, set to zero
38	MI	Multi EU input. Reflects the type of snooping the channel will perform, as programmed by the 'Snoop Type' bit in the descriptor header. 0 Data input snooping by secondary EU disabled. 1 Data input snooping by secondary EU enabled.

Table 14-48. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
39	MO	Multi EU output. Reflects the type of snooping the channel will perform, as programmed by the 'Snoop Type' bit in the descriptor header. 0 Data output snooping by secondary EU disabled. 1 Data output snooping by secondary EU enabled.
40	PR	Request primary EU assignment. The PRI_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_0 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the PRI_GRANT bit. 0 Primary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the Descriptor Header register as long as this bit remains set.
41	SR	Request secondary EU assignment. The SEC_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_1 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the SEC_GRANT bit. 0 Secondary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of secondary EU to the channel. The channel will assert the EU request signal indicated by the Op_1 field in the descriptor header register as long as this bit remains set.
42	PG	Primary EU granted. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
43	SG	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
46	PD	Primary EU done. The PRI_DONE bit reflects the state of the DONE interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.

Table 14-48. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the DONE interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
48–59	ERROR	Crypto-channel error status. This field reflects the error status of the crypto-channel. When a channel error interrupt is generated, this field will reflect the source of the error. The bits in the ERROR field are registered at specific stages in the descriptor processing flow. Once registered, an error can only be cleared only by resetting the crypto-channel or writing the appropriate registers to initiate the processing of a new descriptor. Table 14-52 lists the conditions which can cause a crypto-channel error and how they are represented in the ERROR field.
60–63	PAIR_PTR	Descriptor buffer register length/pointer pair. This field indicates which of the length/pointer pairs are currently being processed by the channel. Table 14-53 shows the meaning of all possible values of the PAIR_PTR field.

Table 14-49 shows the values of the gather state machine.

Table 14-49. G_STATE Field Values

Value	Gather State Machine	Value	Gather State Machine
0x0	idle	0x8	request_bytes_data_tarns_done
0x1	load_4pointers_frm_gather_table	0x9	increase_table_pointer
0x2	load_4pointers_frm_gather_table_done	0xA	update_gather_pointer
0x3	next_bit_set_load_next_gather_table	0xB	gather_table_done
0x4	process_gather_pointer	0xC	gather_error
0x5	request_block_data_trans	0xD	load_next_4pointers_frm_gather_table
0x6	request_block_data_trans_done	0xE–0xF	reserved
0x7	request_bytes_data_tarns	—	—

Table 14-50 shows the values of the scatter state machine.

Table 14-50. S_STATE Field Values

Value	Scatter State Machine:
0x0	idle
0x1	load_4pointers_frm_scatter_table
0x2	load_4pointers_frm_scatter_table_done
0x3	next_bit_set_load_next_scatter_table
0x4	process_scatter_pointer
0x5	request_block_data_trans
0x6	request_block_data_trans_done
0x7	request_bytes_data_tarns

Table 14-50. S_STATE Field Values (continued)

Value	Scatter State Machine:
0x8	request_bytes_data_tarns_done
0x9	increase_table_pointer
0xA	update_scatter_pointer
0xB	scatter_table_done
0xC	scatter_error
0xD	load_next_4pointers_frm_scatter_table
0xE–0xF	reserved

Table 14-51 shows the values of crypto-channel states.

Table 14-51. CHN_STATE Field Values

Value	Channel State	Value	Channel State
0x00	IDLE	0x22	EVALUATE_RESET
0x01	PROCESS_HEADER	0x23	RESET_WRITE_RESET_PRI
0x02	FETCH_DESCRIPTOR	0x24	RESET_RELEASE_PRI_CHA
0x03	CHANNEL_DONE	0x25	RESET_WRITE_RESET_SEC
0x04	CHANNEL_DONE_IRQ	0x26	RESET_RELEASE_SEC_CHA
0x05	CHANNEL_DONE_WRITEBACK	0x27	RESET_CHANNEL
0x06	CHANNEL_DONE_NOTIFICATION	0x28	WRITE_DATASIZE_PRI_POST
0x07	CHANNEL_ERROR	0x29	RESET_RELEASE_ALL
0x08	REQUEST_PRI_CHA	0x2A	RESET_RELEASE_ALL_DELAY
0x09	INC_DATA_PAIR_POINTER	0x2B	REQUEST_SEC_CHA
0x0A	DELAY_DATA_PAIR_UPDATE	0x2C	WRITE_DATASIZE_SEC
0x0B	EVALUATE_DATA_PAIRS	0x2D	WRITE_ICV_SIZE
0x0C	WRITE_RESET_PRI	0x2E	WRITE_SEC_CHA_GO_SNOOPOUT
0x0D	RELEASE_PRI_CHA	0x2F	WRITE_PRI_CHA_GO_SNOOPIN
0x0E	WRITE_RESET_SEC	0x30	WRITE_SEC_CHA_GO_SNOOPIN
0x0F	RELEASE_SEC_CHA	0x31	DELAY_1CYCLE
0x10	PROCESS_DATA_PAIRS	0x33	TRANS_EXTENT_READ
0x11	WRITE_MODE_PRI	0x34	TRANS_EXTENT3
0x12	WRITE_MODE_SEC	0x35	TRANS_EXTENT4
0x13	WRITE_DATASIZE_PRI	0x36	XOR_WRITE_READ_REG
0x14	DELAY_RNGA_DONE	0x37	DELAY_SEC_DONE_TLS
0x15	WRITE_DATASIZE_SEC_SNOOPIN	0x38	MAC_TO_CIPHER
0x16	TRANS_REQUEST_WRITE_SNOOPIN	0x39	MAC_TO_CIPHER_DONE
0x17	DELAY_PRI_SEC_DONE	0x3A	READ_PRI_STATUS
0x18	TRANS_REQUEST_WRITE	0x3C	READ_SEC_STATUS
0x19	WRITE_KEY_SIZE	Others	Reserved
0x1B	DELAY_PRI_DONE		
0x1E	WRITE_DATASIZE_SEC_SNOOPOUT		
0x1F	TRANS_REQUEST_READ_SNOOPOUT		
0x20	DELAY_SEC_DONE		
0x21	TRANS_REQUEST_READ		

Table 14-52 shows the bit positions of each potential error. Multiple errors are possible.

Table 14-52. Crypto-Channel Pointer Status Register Error Field Definitions

Value	Error
48	DOF - Double Fetch Fifo write overflow Error.This bit is set when the Crypto-channel Fetch Fifo is full, SOF is set, and another write has been made to the fetch FIFO. When this bit is set the crypto-channel will stop, and an error interrupt will be activated. The Channel will not start again until a Continue or Reset is given via the CCCR Register. This bit can be cleared by writing '1' to this CCPSR bit.
49	SOF - Single Fetch Fifo write overflow Error. This bit is set when the Channel Fetch Fifo is full and another write has been made to the Fetch Fifo. The Crypto-channel will set this bit and activate an error interrupt. The Crypto-channel continues processing, but the descriptor pointer is lost. The host must clear this bit by writing '1' to this CCPSR bit.
50	MDTE- A Master Data Transfer Error was received from the master bus interface. When the SEC, while acting as a bus master, detects an error, the controller passes the error to the channel in use. The channel halts and activates an interrupt. The channel can only be restarted by writing a '1' to the Continue or Reset bit in the Crypto-Channel Configuration Register, or resetting the whole SEC.
51	Scatter/Gather Data Length Zero Error- A zero length Scatter/Gather data pointer was detected.
52	Fetch Pointer Zero Error- An all zero Fetch Pointer was detected.
53	Illegal descriptor header- Possible causes of an illegal descriptor header are: • Invalid primary EU indicated by op0 field in descriptor header. • Invalid secondary EU indicated by op1 field in descriptor header. • Descriptor type field in descriptor header indicates secondary EU transaction when not in snoop mode
54	Invalid EU assignment request- Indicates the channel has been assigned one or more EUs not requested by the descriptor header.
55	EU error detected. An EU assigned to this channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register.
56	Gather boundary error- Indicates a gather pointer straddles both a primary and secondary EU's data transfer.
57	Gather return/length error- Indicates the total data size covered by a gather link table did not match the total data size from the main descriptor.
58	Scatter Boundary Error- Indicates a scatter pointer straddles both a primary and secondary EU's data transfer.
59	Scatter Return/Length Error- Indicates the total data size covered by a scatter link table did not match the total data size from the main descriptor.

NOTE

EU error bit (ERROR[0]) can only be cleared by first clearing the error source in the assigned EU which caused it to be set.

Table 14-53 shows the possible values of the PTR_DW field in the CCPSR.

Table 14-53. Channel Pointer Status Register PTR_DW Field Values

Value	Error
0x00	Processing Header or pointer dword 0
0x01	Processing pointer dword 1
0x02	Processing pointer dword 2
0x03	Processing pointer dword 3
0x04	Processing pointer dword 4

Table 14-53. Channel Pointer Status Register PTR_DW Field Values (continued)

Value	Error
0x05	Processing pointer dword 5
0x06	Processing pointer dword 6
0x07	Complete (or not yet begun) processing of header dword and pointer dwords
0x08– 0xFF	Reserved

14.6.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The CDPR, shown in Figure 14-63, contains the address of the descriptor which the crypto-channel is currently processing. CPDR, along with the PAIR_PTR in the CCPSR, can be used to determine if a new descriptor can be safely inserted into a chain of descriptors.

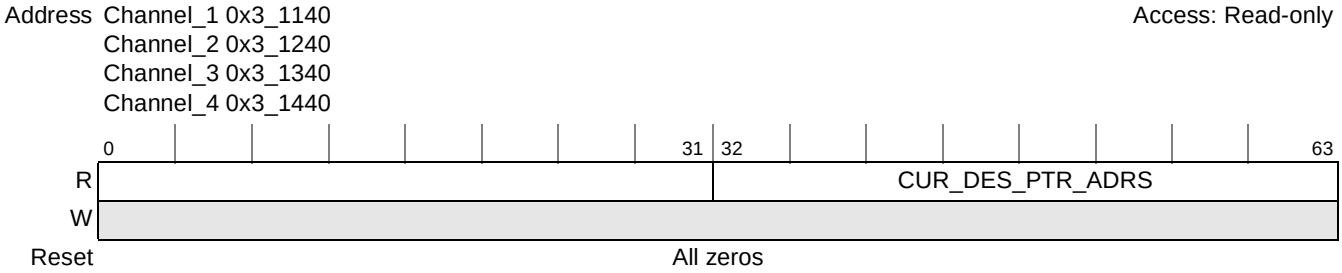


Figure 14-63. Crypto-Channel Current Descriptor Pointer Register

The bits in the current descriptor pointer register perform the functions described in Table 14-54.

Table 14-54. Crypto-Channel Current Descriptor Pointer Register Signals

Bits	Name	Description
0–31	—	Reserved — Set to zero.
32–63	CUR_DES_PTR_ADRS	Current Descriptor Pointer Address. Pointer to system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the channel requests a fetch of a descriptor from the controller. The value from the Fetch FIFO is transferred to the current descriptor pointer register immediately after the fetch is completed. This address will be used as the destination of the writeback of the modified header dword, if header writeback notification is enabled.

14.6.1.4 Fetch FIFO (FF)

Each channel contains a fetch FIFO to store a queue of pointers to descriptors which the channel will process.

The fetch FIFO, shown in Figure 14-64, contains the addresses of the first byte of descriptors to be processed. In typical operation, the host CPU will create a descriptor in memory containing all relevant mode and location information for the SEC, then ‘launch’ the SEC by writing the address of the descriptor to the fetch FIFO.

The fetch FIFO can hold up to 24 descriptor pointers at a time. When the end of the current descriptor is reached, the descriptor pointed to by the next location in the Fetch FIFO will be read to launch the next descriptor. Writing a descriptor pointer to the fetch FIFO while the FIFO is full will result in a single overflow interrupt to advise the user that the descriptor pointer was not successfully written to the fetch FIFO. The channel will continue processing and software can check the fetch FIFO counter in the crypto-channel pointer status register before attempting to re-enqueue the descriptor pointer. If a second descriptor pointer is written to the fetch FIFO before the single the single overflow error is cleared, the channel will generate a double overflow error interrupt and stop processing descriptors. The channel can be restarted by setting the Continue bit in the crypto-channel configuration register, or completely reset by writing the Reset bit in the same register.

The fetch address is written into the FIFO only if the write includes the least significant byte (bits 56–63).

Writing a fetch address of zero to the fetch FIFO causes the channel to generate an error and to stop.

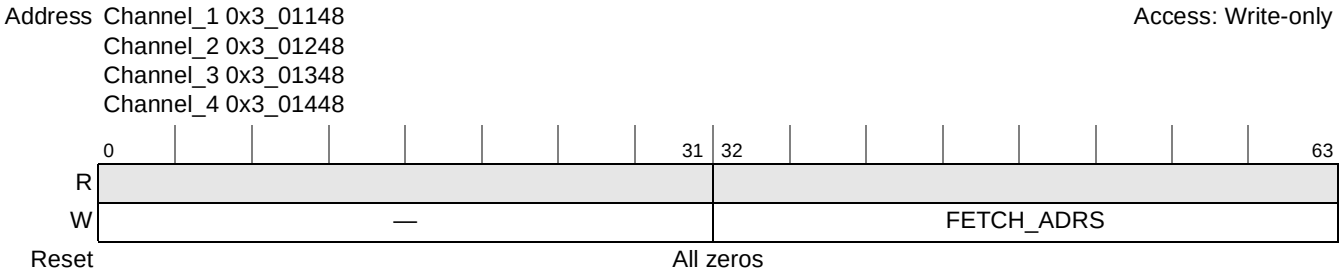


Figure 14-64. Fetch FIFO

Table 14-55 describes the Fetch FIFO fields.

Table 14-55. Fetch FIFO Field Descriptions

Bits	Name	Description
0–31	—	Reserved — Set to zero.
32–63	FETCH ADRS	Fetch Address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

14.6.1.5 Descriptor Buffer (DB)

The descriptor buffer (DB) consists of 8 dword registers (DB0–DB7), and contains the current descriptor being processed by the channel. These registers are read-only, since the descriptor is always fetched from system memory.

For more information about the fields in a descriptor, see Section 14.4.1, “Descriptor Structure.”

	0 15	16	17 23	24 31	32 63
DB0	Header				Reserved
DB1	Length0	J1	Extent0	—	Pointer0
DB2	Length1	J2	Extent1	—	Pointer1
DB3	Length2	J3	Extent2	—	Pointer2
DB4	Length3	J4	Extent3	—	Pointer3
DB5	Length4	J5	Extent4	—	Pointer4
DB6	Length5	J6	Extent5	—	Pointer5
DB7	Length6	J7	Extent6	—	Pointer6
Address	Channel_1 0x3_1180–0x3_11BF, Channel_2 0x3_1280–0x3_12BF, Channel_3 0x3_1380–0x3_13BF, Channel_4 0x3_1480–0x3_14BF				

Figure 14-65. Data Packet Descriptor Buffer

14.6.2 Interrupts

The crypto-channel can assert both DONE and ERROR interrupts to the controller. When the interrupt generation conditions have been met, the crypto-channel will assert the appropriate interrupt. The status of the registered crypto-channel interrupts are available in the controller interrupt status register. The registered interrupts can cleared by writing to the controller interrupt clear register. The crypto-channel does not have an internal interrupt mask bit and error interrupts are always asserted to the controller. The controller can be programmed to mask channel interrupts to the host via its interrupt mask register (IMR). See Section 14.7.2.1, “Interrupt Mask Register (IMR).”

14.6.2.1 Channel Done Interrupt

Whether and when a Channel DONE Interrupt is generated depends on the setting of the Crypto-Channel Configuration Register NT and CDIE bits in the CCCR (see Figure 14-60). Assuming the CDIE (Channel Done Interrupt Enable) is set, the channel will generate an interrupt after every successfully completed descriptor (Notification Type set to Global), or after each successfully completed descriptor with the DN (Done Notification) bit set in the header word of the descriptor.

The Controller will queue multiple Channel Done interrupts if an interrupt is not cleared before the next Channel Done interrupt is issued. If there are multiple interrupts, the controller will re-assert the Channel Done interrupt one cycle after the previous Channel Done interrupt is cleared.

14.6.2.2 Channel Error Interrupt

The Channel Error Interrupt is generated when an error condition occurs during descriptor processing. The channel error interrupt will be asserted as soon as the error condition is detected. The type of error condition is reflected the ERROR field of the Channel Pointer Status Register (CPSR). Refer to Table 14-52 for a complete listing of error types.

14.6.2.3 Channel Reset

Channel reset is asserted when the host sets the RESET bit in the crypto-channel configuration register (CCCR). The effect of software reset on the channel varies according to what the channel is doing when the bit is set:

- If the RESET bit is set while the crypto-channel is requesting a EU assignment from the controller, the crypto-channel will cancel its request by asserting the release output signals. The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.
- If the RESET bit is set after the crypto-channel has been dynamically assigned a EU, the channel will request a write from the controller to set the software reset bit of the EU. A write to reset the secondary (MDEU) EU will also be requested if one has been reserved for snooping. The crypto-channel will then assert the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.

14.7 Controller

The controller within the SEC is responsible for overseeing the operations of the execution units (EUs), the interface to the host processor, and the management of the crypto-channels. The controller interfaces to the host via the master/slave bus interface and to the channels and EUs via internal buses. All transfers between the host and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Arbitrate and control accesses to the bus
- Control the internal bus accesses to the EUs
- Arbitrate and assign EUs to the crypto-channels
- Monitor interrupts from channels and pass to host
- Realign read and write data to the proper byte alignment

14.7.1 Controller Registers

The controller registers are described in detail in the following sections.

14.7.2 EU Assignment Status Register (EUASR)

The EUASR, shown in Figure 14-66, is used to check the assignment status of an EU to a particular channel. When an EU is already assigned, it is inaccessible to any other channel.

A four-bit field (see Table 14-56) indicates the channel to which an EU is assigned.

Offset 0x3_1028 Access: Read-only

	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R	—	AFEU	—	MDEU	—	AESU	—	DEU
W								
Reset	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0

	32–47	48–51	52–55	56–59	60–63
R	—	—	PKEU	—	RNG
W					
Reset	‘0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0

Figure 14-66. EU Assignment Status Register (EUASR)

Table 14-56 shows the EU channel assignments.

Table 14-56. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0xA–0xE	Undefined
0xF	Unavailable

14.7.2.1 Interrupt Mask Register (IMR)

The SEC controller generates the single interrupt output from all possible interrupt sources. These sources can be masked by the interrupt mask register. If unmasked, the interrupt source value, when active, is captured into the interrupt status register. Figure 14-67 shows the bit positions of each potential interrupt source. Each interrupt source is individually unmasked by setting its corresponding bit. At reset, all bits are masked. The bit fields are described in Table 14-57.

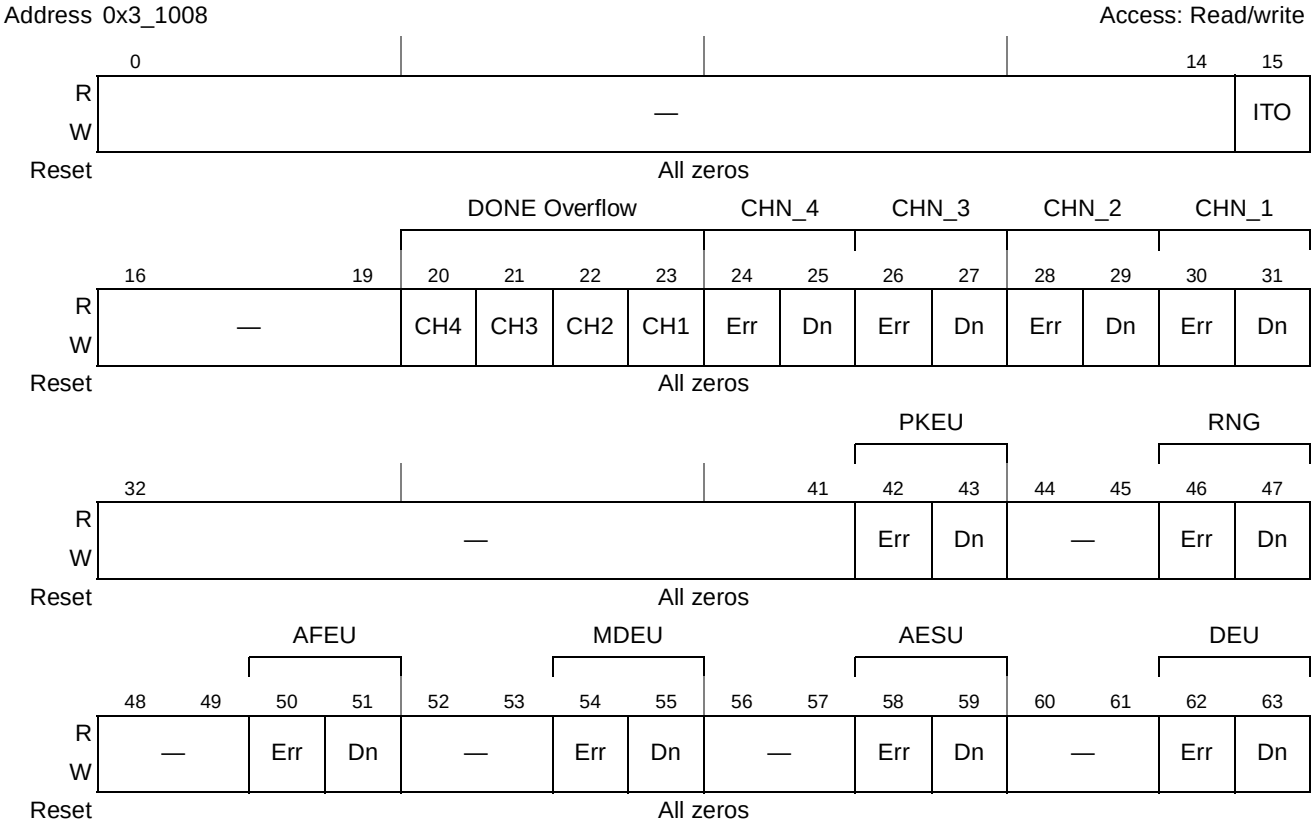


Figure 14-67. Interrupt Mask Register

14.7.2.2 Interrupt Status Register (ISR)

The ISR contains fields representing all possible sources of interrupts. It is cleared either by a reset or by writing the appropriate active ICR bits. Figure 14-68 shows the bit positions of each potential interrupt source. The bit fields are described in Table 14-57.

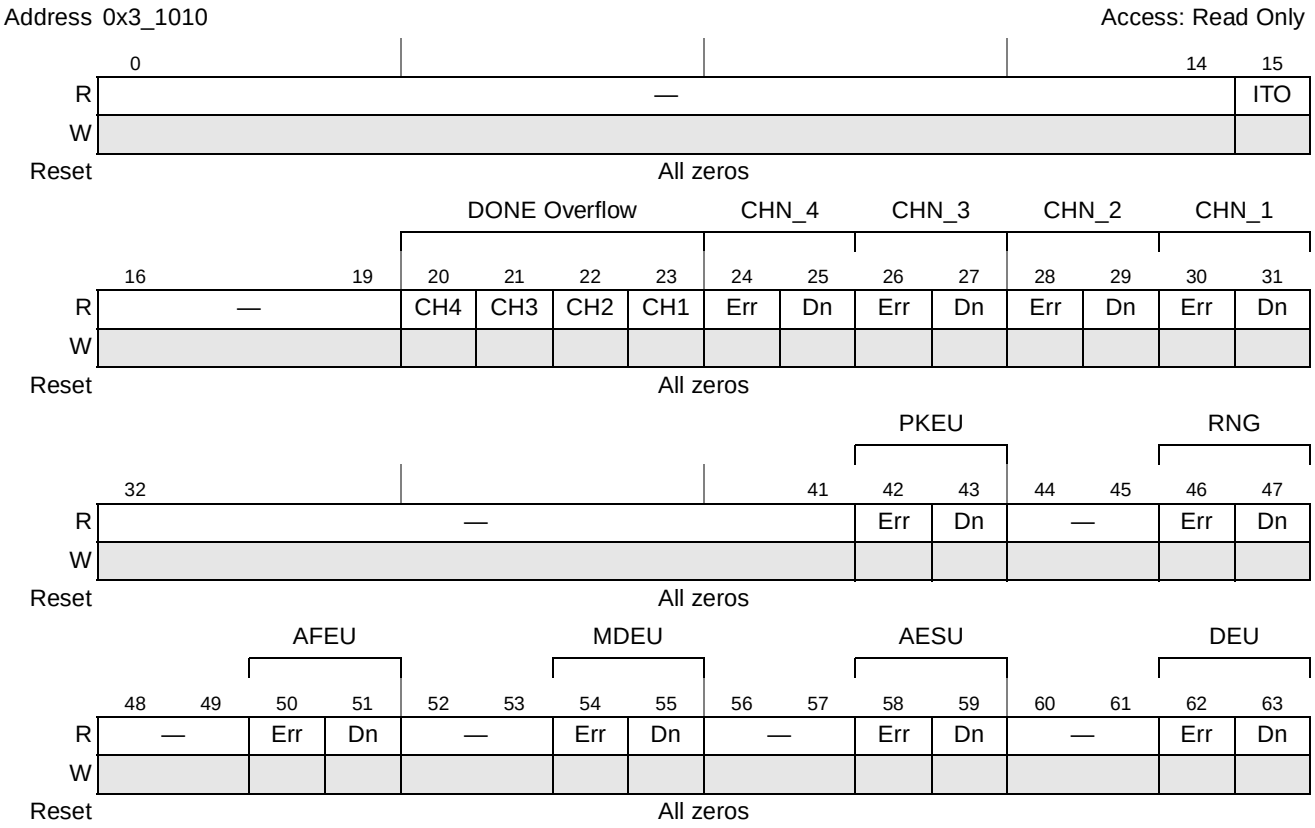


Figure 14-68. Interrupt Status Register

14.7.2.3 Interrupt Clear Register (ICR)

ICR provides a means of clearing the interrupt status register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output pin $\overline{\text{IRQ}}$ (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 14-69 shows the bit positions of each interrupt source that can be cleared by ICR. The complete bit definitions for ICR can be found in Figure 14-69. ICR fields are described in Table 14-57.

When an ICR bit is written, it will automatically clear itself one cycle later. That is, it is not necessary to write a ‘0’ to a bit position which has been written with a ‘1.’

NOTE

Interrupts are registered and sent based upon the conditions which cause them. If the cause of an interrupt is not removed, the interrupt will return a few cycles after it has been cleared using the ICR.

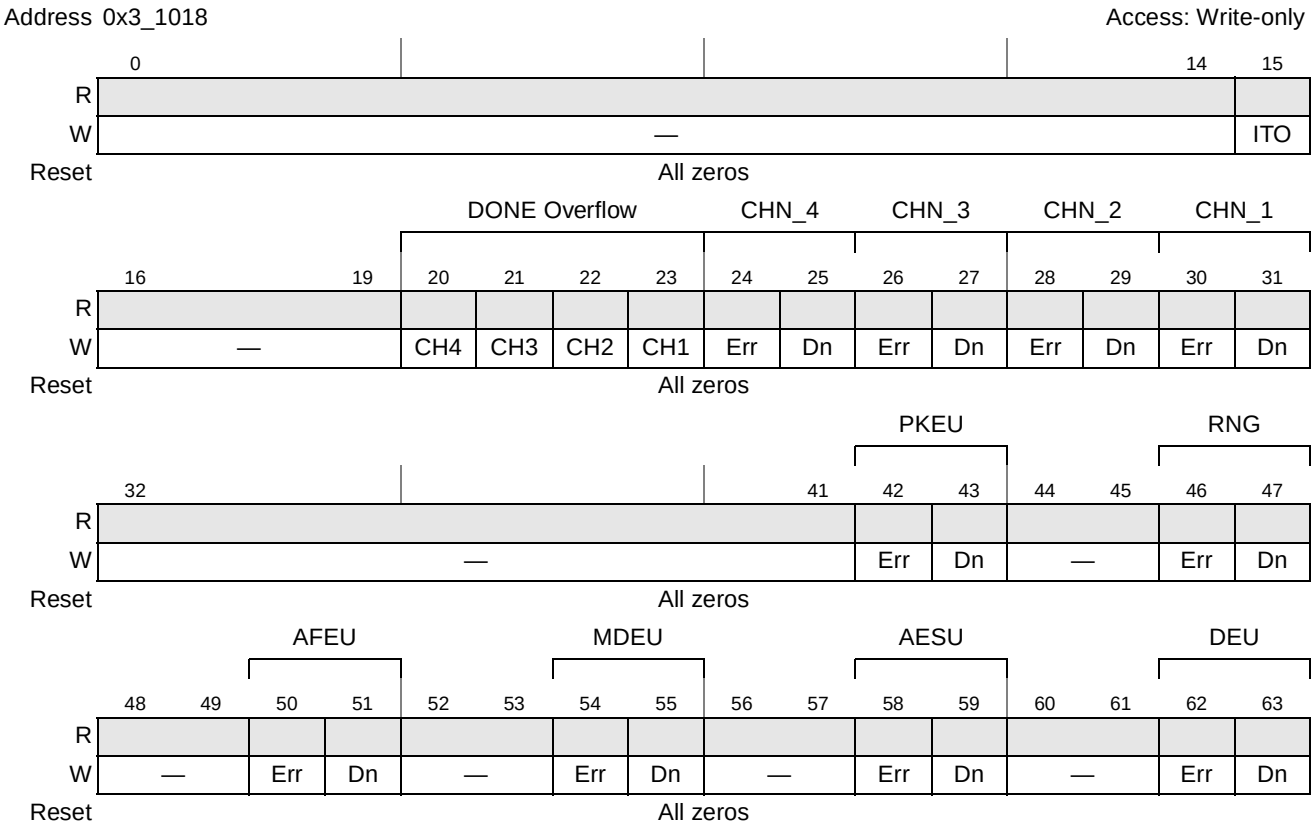


Figure 14-69. Interrupt Clear Register

Table 14-57 describes the interrupt mask, status, and clear register fields.

Table 14-57. Interrupt Mask, Status, and Clear Register Fields

Bits	Name	Description
15	ITO	Internal Time Out 0 No Internal Time Out 1 An Internal Time Out was detected The Internal Time Out interrupt is triggered by the controller if a slave access to an SEC register does not result in successful data transfer within 16 clock cycles. With ITO enabled the SEC controller terminates the transaction and signals and interrupt.
20–23	Done Overflow	Done overflow 0 No Done Overflow 1 Done Overflow Error. Indicates that more than 15 Done interrupts were queued from the interrupting channel without an interrupt clear.
Multiple	CHN_Err_Dn	Each of the 4 channels has Error & Done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.

Table 14-57. Interrupt Mask, Status, and Clear Register Fields (continued)

Bits	Name	Description
Multiple	EU_Err_Dn	Each of the execution units has Error & Done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.
0–14,16–19, 32–41,44–45, 48–49,52–53, 56–57,60–61	—	Reserved, set to zero.

14.7.2.4 ID Register

The read-only ID register, shown in Figure 14-70, contains the same value as the IP block revision register (see Section 14.7.2.5, “IP Block Revision Register”). In updating existing software, use of this address is likely to be most convenient, since it is the same address used for version information in previous revisions of the SEC.

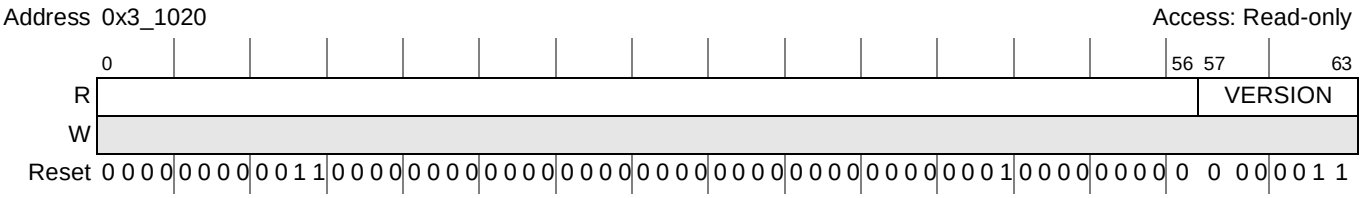


Figure 14-70. ID Register

14.7.2.5 IP Block Revision Register

The read-only IP block revision register, shown in Figure 14-71, contains a 64-bit value that uniquely identifies the version of the SEC 2.4. The value of this register is 0x0030_0000_0001_0003.

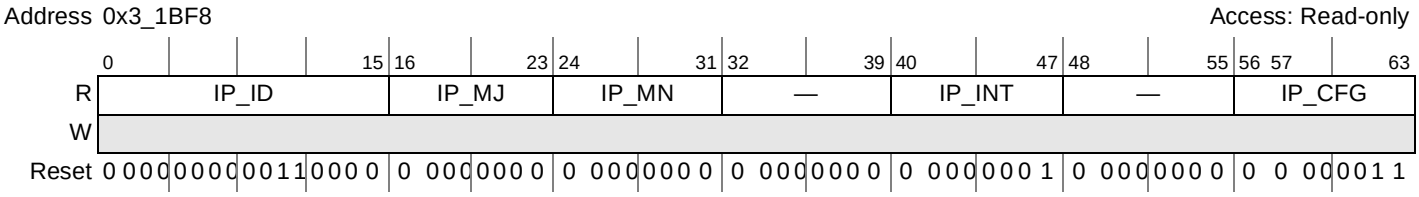


Figure 14-71. IP Block Revision Register

Table 14-58 describes the fields of the IP block revision revision register.

Table 14-58. IP Block Revision Register Fields

Bits	Name	Description
0–15	IP_ID	IP block identifier
16–23	IP_MJ	IP major revision number

Table 14-58. IP Block Revision Register Fields (continued)

Bits	Name	Description
24–31	IP_MN	IP minor revision number
32–39	—	Reserved
40–47	IP_INT	IP block integration options
48–55	—	Reserved
56–63	IP_CFG	IP block configuration options

14.7.2.6 Master Control Register (MCR)

The MCR, shown in Figure 14-72, controls certain functions in the controller and provides a way for software to reset the SEC.

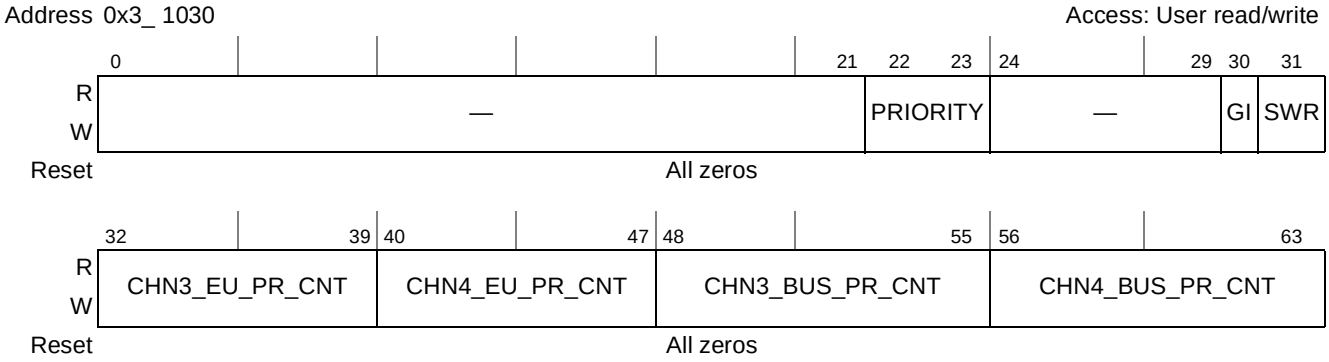


Figure 14-72. Master Control Register

Table 14-59 describes the master control register signals.

Table 14-59. Master Control Register Signals

Bits	Name	Description
0–21	—	Reserved
22–23	Priority	Priority on Master Bus. The setting of these bits determines the transaction priority level the SEC asserts to the device internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software may change the SEC priority level in realtime. 00 - Lowest Priority (default) 01 - Next Lowest Priority 10 - Next Highest Priority 11 - Highest Priority
24–29	—	Reserved
30	GI	Global Inhibit. 0 Master will always drive GBL_B active (low), meaning that the transactions are global and cache snooping is needed in order to enforce coherency. 1 Master will always drive GBL_B inactive (high).

Table 14-59. Master Control Register Signals (continued)

Bits	Name	Description
31	SWR	Software Reset. Writing 1 to this bit will cause a global software reset. Upon completion of the reset, this bit will be automatically cleared. 0 Don't reset 1 Global Reset **Note:** Warning: Certain SEC interrupts are not fully cleared by writing this bit. If SEC interrupts are pending, it is recommended that the user set this bit twice (two consecutive writes) to completely reset the SEC.
32–39	CHN3_EU_PR_CNT	Channel 3 EU Priority Counter. This counter is used by the controller to determine when Channel 3 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. **Note:** If set to zero, the CHN4_EU_PR_CTR must also be set to zero, and the controller will assign EUs on a pure round robin basis. If set to non-zero, CHN4_EU_PR_CTR must also be set to a different, non-zero value.
40–47	CHN4_EU_PR_CNT	Channel 4 EU Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. **Note:** If set to zero, the CHN3_EU_PR_CTR must also be set to zero, and the controller will assign EUs on a pure round robin basis. If set to non-zero, CHN3_EU_PR_CTR must also be set to a different, non-zero value.
48–55	CHN3_BUS_PR_CNT	Channel 3 Bus Priority Counter.This counter is used by the controller to determine when Channel 3 has been denied access to the bus long enough to warrant immediate elevation to top priority. **Note:** If set to zero, the CHN4_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN4_BUS_PR_CTR must also be set to a different, non-zero value.
56–63	CHN4_BUS_PR_CNT	Channel 4 Bus Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to a needed on-chip resource long enough to warrant immediate elevation to top priority. **Note:** If set to zero, the CHN3_BUS_PR_CTR must also be set to zero, and the controller will assign access to the bus on a pure round robin basis. If set to non-zero, CHN3_BUS_PR_CTR must also be set to a different, non-zero value.

14.7.2.7 EU Access

Assignment of a EU function to a channel is done dynamically. With dynamic assignment, the channel requests a EU function, the controller checks to see if the requested EU function is available, and if it is, the controller grants the channel assignment of the EU.

If an EU is available for a channel when requested, the controller will assert the grant signal pertaining to the request from the channel. The grant signal will remain asserted until the channel is done and releases the EU.

14.7.2.8 Multiple EU Assignment

In some cases, a channel may request two EUs. The channel will do this by first requesting the primary EU, then requesting the secondary EU. Once the controller has granted both EUs, this channel is then capable of requesting that the secondary EU snoop the bus. Snooping is described in Section 14.8.3, "Snooping by Caches".

In all cases, the controller assigns the primary EU to a requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing any grants to a channel which is requesting two EU functions.

14.7.2.9 Multiple Channels

Since there are multiple channels in the SEC, the controller must arbitrate for access to the execution units. To accomplish this, the controller implements an arbiter for each channel.

Each arbiter acts on either a weighted priority-based or round-robin scheme, depending on the values of CHN3_EU_PR_CNT and CHN4_EU_PR_CNT. If both CHN3_EU_PR_CNT and CHN4_EU_PR_CNT are set to a non-zero value, the arbiter will implement the weighted priority scheme. Otherwise, the arbitration will be round-robin. Setting only one of the CHN_EU_PR_CNT fields to a non-zero value causes unpredictable operation.

14.7.2.10 Priority Arbitration

When arbitrating on the priority scheme, the priority will be as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHN3_EU_PR_CNT or CHN4_EU_PR_CNT expired
- Channel 3—Third priority, unless CHN4_EU_PR_CNT expired
- Channel 4—Lowest priority, until CHN4_EU_PR_CNT expired

For channels 1–4, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, the CHN3_EU_PR_CNT and CHN4_EU_PR_CNT fields are implemented in the Master Control Register. The value of these fields determines how many times channel 3 or channel 4 can be refused access to an EU in favor of a higher priority channel. A counter is implemented in the arbiter for each of these entities. When the channel has lost arbitration the number of times specified in its CHN_EU_PR_CNT field, then that channel has the 2nd highest priority when the requested EU becomes available. CHN1 always has the highest priority, but it cannot make back to back requests, so the 2nd highest priority channel will be serviced upon completion of the current CHN1 operation.

It is permissible for the CHN_EU_PR_CNT values to be different from the CHN_BUS_PR_CNT values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice-versa.

14.7.2.11 Round-Robin Snapshot Arbiters

The controller implements seven ‘snapshot’ arbiters, one for each EU function, and one for the bus. Each arbiter takes a snapshot of the requests for its function. If there are requests, then the arbiter satisfies those requests via a round-robin scheme as the resource becomes available. When all requests have been satisfied, the arbiter takes another snapshot.

14.8 Bus Interface

The controller in the SEC (refer to Section 14.7, “Controller”) has the ability to be a bus master or a slave. This means that the controller can issue read and write commands to the bus, and it can also be written to and read from by the host.

The controller is the sole bus master in the SEC. All other modules are slave only devices. A channel may request access to system resources including the bus. In these cases, the channel must provide the starting address of the transfer for the bus(es) requested. All subsequent addresses are generated by the controller. All addresses will be sequential.

14.8.1 Bus Access

The controller attempts to maximize bus utilization by grouping outstanding bus requests from the channels by request type. The Controller will push all write requests to the Bus Interface, followed by all read requests, then repeat. Within a request type, the Controller will granted bus access via the same scheme that is used for granting EUs. When the CHN_BUS_PR_CNT values of both channel 3 and 4 are set to zero, round robin operation is in effect. In this case, the snapshot arbiter samples the requests for the bus, then grants those requests as the bus becomes available. For example, if channels 1, 2, and 4 are requesting bus access at a given time, the snapshot arbiter will register the three requests and ignore further requests. The buses will be granted to channel 1 until its transfer is completely satisfied. Then the buses will be granted to channel 2 until channel 2’s transfer is completely satisfied. Finally, the buses will be granted to channel 4 until that transfer is completely satisfied. Then another snapshot of requests will be taken. Refer to Section 14.7.2.10, “Priority Arbitration,” for more information.

14.8.1.1 Master Read

The sequence for master read access is as follows:

1. Channel asserts its bus read request.
2. Channel furnishes address and transfer length.
3. Controller acknowledges request to channel.
4. Controller asserts request to Master interface.
5. Controller waits for bus read to begin.
6. When bus read begins, controller receives data from the master interface and performs a write to the appropriate internal address using the address supplied by the channel. Data may be realigned byte-wise by the controller if either:
   - the read did not begin on a 32-bit word boundary, or
   - the previous write to an execution unit’s input FIFO did not end on a 32-bit word boundary
7. Transfer continues until the bus read is completed and the controller has written all data to the appropriate internal address. The master interface will continue making bus requests until the full data length has been read.

14.8.1.2 Master Write

Master writes are performed by transferring data from one of the EUs to the output FIFO in the controller, then transferring the data from the FIFO to the bus when the bus is granted to the controller. The sequence for a master bus write access is as follows:

1. Channel requests the bus from controller.
2. Channel furnishes address, transfer length.
3. Controller acknowledges request to channel.
4. Controller loads the write data into its FIFO, and waits for the bus to become available.
5. When the bus becomes available, controller writes data from its FIFO to the master interface.
6. Transfer continues until the bus write is completed and the controller has read all data from the appropriate internal address. The master interface will continue making bus requests until the full data length has been written.

It is probable that multiple bus bursts will be required to complete any given request. When a channel has been granted access to the bus, no other internal SEC requests to the bus will be acknowledged until that transfer has been fully satisfied; i.e., all bytes have been transferred.

14.8.1.2.1 Slave Access

As a bus slave, the controller simply responds to read and write commands from the bus. When it receives a write command, the controller takes the data from the slave interface and sends it to the internal location indicated by the address. For a read, the controller goes to the internal location and fetches the requested data from the specified address. The SEC internal memory space must be accessed modulo-4 boundaries to avoid invalid data or unpredictable operation.

14.8.2 Bus Arbitration Priority

Transaction priority is configured for all channels in the SEC master control register (MCR[Priority]) as shown in Figure 14-72. The SEC does not dynamically adjust its transaction priority, however system software can dynamically adjust SEC transaction priority, with the change in priority taking effect immediately.

14.8.3 Snooping by Caches

If defined as global, SEC transactions can be snooped by the MPC8360E cache. This definition is programmed in the master control register MCR[GI]. See 14.7.2.6, “Master Control Register (MCR),” for more details. Note that SEC transactions are defined as global by default.

14.8.4 Interrupts

The SEC generates a single interrupt to the programmable interrupt controller. Interrupts from the SEC are reported to the CPU if the mask bit in SIMSR_H[SEC] is set.

The user can control which events cause an interrupt by configuring the SEC interrupt mask register. These events are as follows:

- Done (of a channel or an execution unit)
- Error (of a channel or an execution unit)

When the user detects an interrupt request from the SEC, it should further read the SEC interrupt status register (ISR) to determine the interrupt source. To clear an interrupt, the user should write 1 to the corresponding bits in the SEC interrupt clear register (ICR).

Events may be further masked per channel by setting or clearing the related fields in the crypto-channel configuration registers. It is suggested that the user leave channel interrupts unmasked, while masking the interrupts from the EUs. Errors or Done signals coming from the EUs eventually cause the channel to signal an ERROR or DONE interrupt. Clearing an interrupt before eliminating the condition that caused the interrupt will cause the interrupt to be asserted again a few cycles later.

14.9 Power–Saving Mode

The SEC can be disabled by clearing SCCR[ENCCM]. See Section 4.5.2.3, “System Clock Control Register (SCCR),” for more information.

Chapter 15
I^{2}C Interfaces

This chapter describes the two inter-IC (IIC or I^{2}C) bus interfaces implemented on this device. Note that for most intents, the I^{2}C interfaces are identical and are described as a single generic controller. Where necessary, differences between the two controllers are noted.

15.1 Introduction

The inter-IC (IIC or I^{2}C) 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 I^{2}C interface.

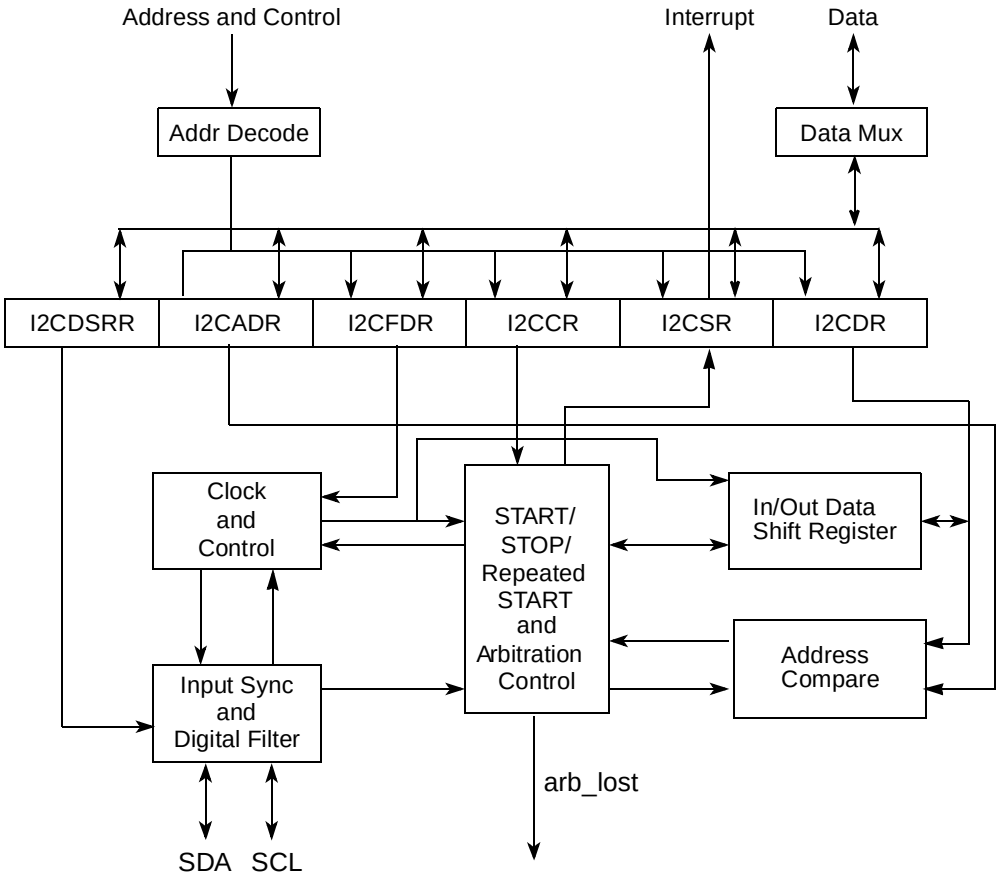


Figure 15-1. I^{2}C Block Diagram

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

Each I²C interface includes the following features:

- Two-wire interface
- Multiple-master operational
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

15.1.2 Modes of Operation

The I²C unit on this device can operate in one of the following modes:

- Master mode. The I²C initiates a transfer, generates clock signals, and terminates a transfer. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode. The I²C is addressed by an I²C master. The module must be enabled before a START condition from an I²C master is detected.
- Interrupt-driven byte-to-byte data transfer. When successful slave addressing is achieved (and SCL*n* returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode. I²C1 controller supports boot sequencer mode. This mode can be used to initialize the configuration registers in the device after the I²C module is initialized. Boot sequencer mode is selected using the BOOTSEQ field in the reset configuration word high. Note that the hard-coded reset configuration word high value is boot sequencer mode disabled. This mode is not supported by I²C2 controller.
- Reset configuration load (I²C1 only). In this mode, the I²C1 interface loads the reset configuration words from an EEPROM at a specific calling address while the rest of the device is in the reset state ($\overline{\text{HRESET}}$ asserted). Once the reset configuration words are latched inside the device, I²C1 is reset until $\overline{\text{HRESET}}$ is negated. After $\overline{\text{HRESET}}$ is negated, the device may be initialized using boot

sequence mode according to the BOOTSEQ field in the reset configuration word. See Section 4.3.3.2, “Loading from I2C EEPROM.”

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*n* and SCL*n* signals, described in Table 15-1, for data transfer. Note that the signal patterns driven on SDA*n* represent address, data, or read/write information at different stages of the protocol.

Table 15-1. I²C Interface Signal Description

Signal Name	Idle State	I/O	State Meaning
Serial Clock (SCL1, SCL2)	High	I	When the I²C module is idle or acts as a slave, SCL*n* defaults as an input. The unit uses SCL*n* to synchronize incoming data on SDA*n*. The bus is assumed to be busy when SCL*n* is detected low.
		O	As a master, the I²C module drives SCL*n* along with SDA*n* when transmitting. As a slave, the I²C module drives SCL*n* negates for data pacing.
Serial Data (SDA1, SDA2)	High	I	When the I²C module is idle or in a receiving mode, SDA*n* defaults as an input. The unit receives data from other I²C devices on SDA*n*. The bus is assumed to be busy when SDA*n* is detected low.
		O	When writing as a master or slave, the I²C module drives data on SDA*n* synchronous to SCL*n*.

15.2.2 Detailed Signal Descriptions

SDA*n* and SCL*n*, described in Table 15-2, serve as a communication interconnect with other devices. All devices connected to these signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the hardware specifications for electrical characteristics.

Table 15-2. I²C Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description
SCL1, SCL2	I/O	Serial clock. Performs as an input when the device is programmed as an I²C slave. SCL*n* 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*n*. The bus is assumed to be busy when this signal is detected low.
SDA1, SDA2	I/O	Serial data. Performs as an input when the device is in a receiving mode. SDA*n* also performs as an output signal when the device is transmitting (as an 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*n* 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	I2C1ADR—I²C1 address register	R/W	0x00	15.3.1.1/15-5
0x0_3004	I2C1FDR—I²C1 frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3008	I2C1CR—I²C1 control register	R/W	0x00	15.3.1.3/15-6
0x0_300C	I2C1SR—I²C1 status register	R/W	0x81	15.3.1.4/15-8
0x0_3010	I2C1DR—I²C1 data register	R/W	0x00	15.3.1.5/15-9
0x0_3014	I2C1DFSRR—I²C1 digital filter sampling rate register	R/W	0x10	15.3.1.6/15-10
0x0_301C–0x0_30FF	Reserved, should be cleared	—	—	—
0x0_3100	I2C2ADR—I²C2 address register	R/W	0x00	15.3.1.1/15-5
0x0_3104	I2C2FDR—I²C2 frequency divider register	R/W	0x00	15.3.1.2/15-5
0x0_3108	I2C2CR—I²C2 control register	R/W	0x00	15.3.1.3/15-6
0x0_310C	I2C2SR—I²C2 status register	R/W	0x81	15.3.1.4/15-8
0x0_3110	I2C2DR—I²C2 data register	R/W	0x00	15.3.1.5/15-9
0x0_3114	I2C2DFSRR—I²C2 digital filter sampling rate register	R/W	0x10	15.3.1.6/15-10
0x0_311C–0x0_31FF	Reserved, should be cleared	—	—	—

15.3.1 Register Descriptions

This section describes the I²C registers in detail. Note that reserved bits should always be written with the value they return when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero. This does not apply to the I²C*n* data register (I2C*n*DR).

15.3.1.1 I²C*n* Address Register (I2C*n*ADR)

Figure 15-2 shows the I2C*n*ADR register, which contains the address to which the I²C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I²C module is in master mode.

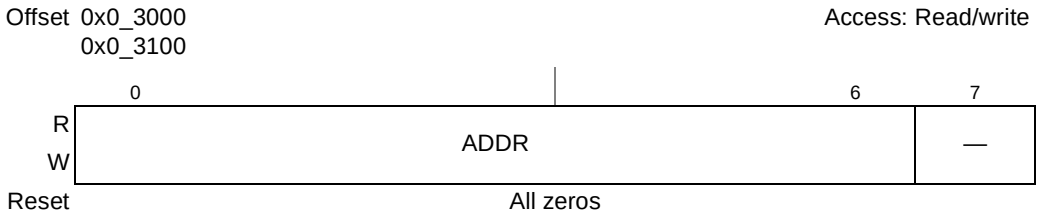


Figure 15-2. I²C*n* Address Register (I2C*n*ADR)

Table 15-4 describes the bit settings of I2C*n*ADR.

Table 15-4. I2C*n*ADR 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 I2C*n*SR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved, should be cleared

15.3.1.2 I²C*n* Frequency Divider Register (I2C*n*FDR)

Figure 15-3 shows the bits of the I²C*n* frequency divider register.

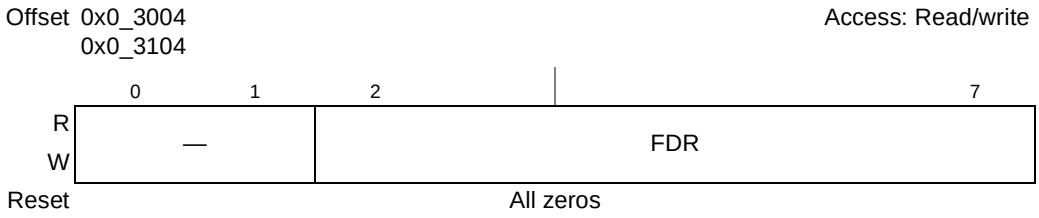


Figure 15-3. I²C*n* Frequency Divider Register (I2C*n*FDR)

Table 15-5 describes the bit settings of I2C*n*FDR. It also maps I2C*n*FDR[FDR] to the clock divider values. Although it describes the ratio between the I²C controller internal clock and SCL

Table 15-5. I2C*n* 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*n* is equal to the I²C*n* 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].

The I²C controller internal clock frequency is equal to the CSB clock frequency for both I²C1 and I²C2.

15.3.1.3 I²C*n* Control Register (I2C*n*CR)

Figure 15-4 shows the I²C*n* control register.

Offset 0x0_3008 Access: Mixed
0x0_3108

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK		—	BCST
W						RSTA		
Reset	All zeros							

Figure 15-4. I²C*n* Control Register (I2C*n*CR)

Table 15-6 describes the I2C*n*CR bit settings.

Table 15-6. I2C*n*CR 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 I2C*n*SR[MIF] is also set.
2	MSTA	Master/slave mode START 0 On a transition to zero, a STOP condition is generated and the mode changes from master to slave. Cleared without generating a STOP condition when the master loses arbitration. 1 When MSTA changes from zero to one, a START condition is generated on the bus and master mode is selected.
3	MTX	Transmit/receive mode select. Selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2C*n*SR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always high. MTX is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. Specifies the value driven onto the SDA*n* line during acknowledge cycles for both master and slave receivers. The value of this bit applies only when the 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*n*) is sent out to the bus at the 9th clock bit after receiving one byte of data. 1 No acknowledge signal response (high value on SDA*n*) is sent.
5	RSTA	Repeated START. Note that this bit is not readable, which means if a read is performed to RSTA, a zero value is returned. 0 No START condition is generated 1 Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration.
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*n* Status Register (I2C*n*SR)

I2C*n*SR is shown in Figure 15-5.

Offset 0x0_300C
0x0_310C

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*n* Status Register (I2C*n*SR)

Table 15-7 describes the bit settings of the I2C*n*SR.

Table 15-7. I2C*n*SR 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 I2C*n*DR is read in receive mode or when I2C*n*DR 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 I2C*n*ADR matches the calling address or when the calling address is the broadcast address and broadcast mode is enabled (I2C*n*CR[BCST] is set), this bit is set. The processor is interrupted if I2C*n*CR[MIEN] is set. Next, the processor must check the SRW bit and set I2C*n*CR[MTX] accordingly. Writing to the I2C*n*CR 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 I2C*n*CR 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.

Table 15-7. I2C*n*SR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2C*n*CR[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 I2C*n*ADR 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*n* Data Register (I2C*n*DR)

The I2C*n* data register is shown in Figure 15-6.

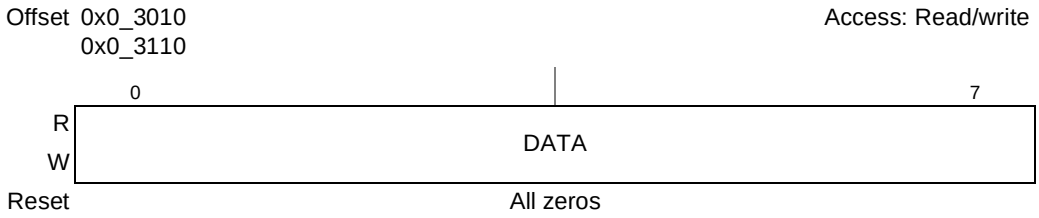


Figure 15-6. I2C*n* Data Register (I2C*n*DR)

Table 15-8 shows the bit descriptions for I2C*n*DR.

Table 15-8. I2C*n*DR Field Description

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 I2C*n*DR. 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 very first read is always a dummy read.

15.3.1.6 Digital Filter Sampling Rate Register (I2C*n*DFSRR)

I2C*n*DFSRR is shown in Figure 15-7.

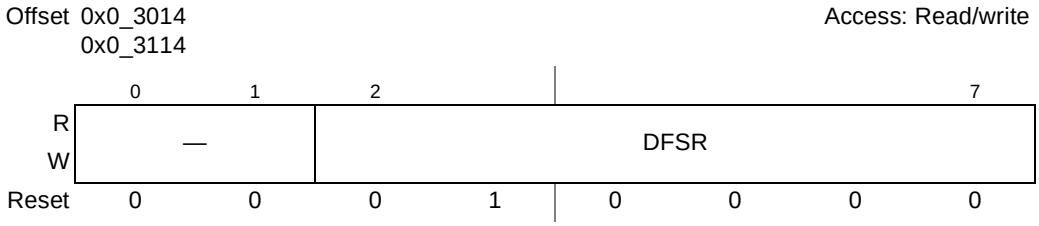


Figure 15-7. I²C*n* Digital Filter Sampling Rate Register (I2C*n*DFSRR)

Table 15-9 shows the I2C*n*DFSRR field descriptions.

Table 15-9. I2C*n*DFSRR 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 I2C*n*DFSRR 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.

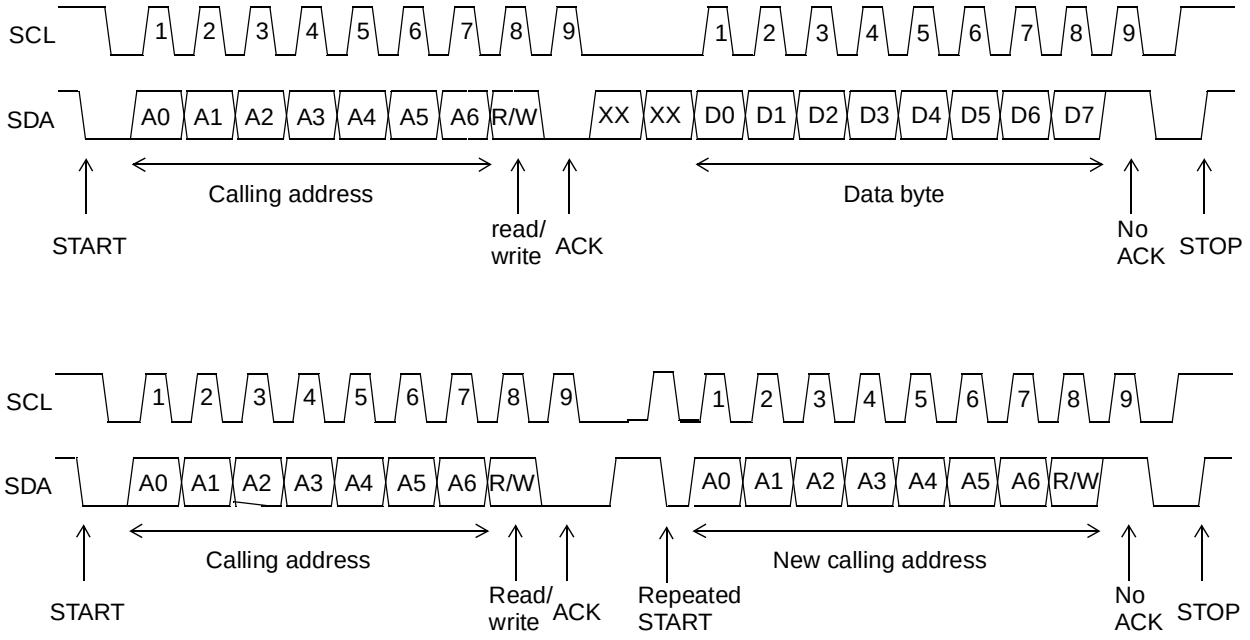


Figure 15-8. I²C Interface Transaction Protocol

15.4.1.1 START Condition

When the I²C bus is not engaged (both SDA*n* and SCL*n* lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 15-8, a START condition is defined as a high-to-low transition of SDA*n* while SCL*n* is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets I2C*n*CR[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*n* signal at the 9th clock) as shown in Figure 15-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL*n* returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

The I²C module responds to a general call (broadcast) command when I2C*n*CR[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 I2C*n*DR with I2C*n*CR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

Each data byte is 8 bits long. Data bits can be changed only while SCL*n* is low and must be held stable while SCL*n* is high, as shown in Figure 15-8. There is one clock pulse on SCL*n* for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling SDA*n* low at the 9th clock. Therefore, one complete data byte transfer takes 9 clock pulses. Several bytes can be transferred during a data transfer session.

If the slave receiver does not acknowledge the master, the SDA*n* line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA*n* line for the master to generate a STOP or a START condition.

15.4.1.3 Repeated START Condition

Figure 15-8 shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

15.4.1.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA*n* signal while SCL*n* is high. For more information, see Figure 15-8. Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2C*n*CR[MSTA].

As described in Section 15.4.1.3, “Repeated START Condition,” the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

15.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in the I²C module.

15.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows (see Figure 15-8):

- START conditions are detected when an SDA*n* fall occurs while SCL*n* is high.
- STOP conditions are detected when an SDA*n* rise occurs while SCL*n* is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition and idle upon the detection of a STOP condition.

15.4.1.5.2 Control Transfer—Implementation Details

The I²C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL*n* output is pulled low as determined by the internal clock generated in the clock module. The SDA*n* output can change only at the midpoint of a low cycle of the SCL*n*, unless it is performing a START, STOP, or repeated START condition. Otherwise, the SDA*n* output is held constant.

SDA*n* is negated when one or more of the following conditions are true:

- Master mode
  - Data bit (transmit)
  - ACK bit (receive)
  - START condition
  - STOP condition
  - Repeated START condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - ACK bit (receive)

The SCL*n* signal corresponds to the internal SCL*n* signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Repeated START condition begin
  - Repeated START condition end

- Slave mode
  - — Address cycle
  - — Transmit cycle
  - — ACK cycle

15.4.1.6 Address Compare—Implementation Details

The address compare block determines whether a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The following address comparisons are performed:

- Whether a broadcast message has been received, to update I2C*n*SR
- Whether the module has been addressed as a slave, to update I2C*n*SR and to generate an interrupt
- Whether the address transmitted by the current master matches the general broadcast address

15.4.2 Arbitration Procedure

The I^2C 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 I^2C module) determines the bus clock—the low period is equal to the longest clock-low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA*n* while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA*n* line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I^2C unit sets I2C*n*SR[MAL] to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

If the I^2C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I^2C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I^2C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately causes in the current bus master to lose arbitration, after which bus operations return to normal.

15.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA*n* line while attempting to drive a 1, tries to generate a START or repeated START at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2C*n*SR[MAL] is set) under the following conditions:

- SDA*n* samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA*n* samples low when the master drives high during a data-receive cycle of the acknowledge (ACK) bit (receive).

- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.
- A repeated START condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

Note that the I²C module does not automatically retry a failed transfer attempt.

15.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL*n* low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL*n* line.

15.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
  - Transmit slave address after START condition
  - Transmit slave address after repeated START condition
  - Transmit data
  - Receive data
- Slave mode
  - Transmit data
  - Receive data
  - Receive slave address after START or repeated START condition

15.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL*n* line, a high-to-low transition on the SCL*n* line affects all devices connected on the bus. The devices begin counting their low period when the master negates the SCL*n* line. After a device has negated SCL*n*, it holds the SCL*n* line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of SCL*n* if another device is still within its low period. Therefore, SCL*n* is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low periods, SCL*n* is released and asserted. Then there is no difference between the devices' clocks and the state of SCL*n*, and all the devices begin counting their high periods. The first device to complete its high period negates SCL*n* again.

15.4.4.2 Input Synchronization and Digital Filter

The following sections describes synchronization of the input signals and the filtering of SCL*n* and SDA*n* in detail.

15.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL*n* and SDA*n* signals to the system clock and detects transitions of these signals.

15.4.4.2.2 Filtering of SCL*n* and SDA*n* Lines

The SCL*n* and SDA*n* inputs are filtered to eliminate noise. Three consecutive samples of the SCL*n* and SDA*n* lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

The sampling rate is equal to a binary value stored in the frequency register I2CDFSRR. The duration of the sampling cycle is controlled by a down counter. This allows a software write to the I2CDFSRR to control the filtered sampling rate.

15.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven SCL*n* low, the slave can drive SCL*n* low for the required period and then release it. If the slave SCL*n* low period is greater than the master SCL*n* low period, the resulting SCL*n* low period is extended.

15.4.5 Boot Sequencer Mode

Boot sequencer mode is selected at power-on reset by the BOOTSEQ field of the high-order reset configuration word. If boot sequencer mode is selected, the I²C module communicates with one or more EEPROMs through the I²C interface. EEPROMs can be programmed to initialize one or more configuration registers. Note that as described in Section 4.3.2.2.3, “Boot Sequencer Configuration,” the default value for BOOTSEQ is 0b00, which corresponds to the I²C boot sequencer being disabled at power-up.

Boot sequencer mode also supports an extension of the standard I²C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes. This extended addressing mode is selectable using a different encoding in the BOOTSEQ field of the high-order reset configuration word (see Section 4.3.2.2.3, “Boot Sequencer Configuration.”) In this mode, only one EEPROM device can be used and the maximum number of registers is limited by the size of the EEPROM.

If the standard I²C interface is used, the I²C module addresses the first EEPROM, and reads 256 bytes. Then it issues a repeated start and addresses the next EEPROM address. This sequence continues until the CONT bit is cleared. If the last register is not detected before wrapping back to the first address, an error condition is detected. In other words, if the CONT bit for not cleared on the final 7 bytes, an error condition is detected, causing the I²C controller to hang. The I²C module continues to read from the EEPROM as long as the continue (CONT) bit is set in the EEPROM. The CONT bit resides in the address/attributes

field that is transferred from the EEPROM, as described in Section 15.4.5.2, "EEPROM Calling Address." There should be no other I²C traffic when the boot sequencer is active.

15.4.5.1 Using the Boot Sequencer for Reset Configuration

The reset configuration word can be loaded by using the I²C boot sequencer. See Section 4.3.2.2.3, "Boot Sequencer Configuration."

Note that this usage does not prevent using the I²C boot sequencer to initiate the device in the normal functional mode, after reset state has completed. However, an I²C serial EEPROM of extended addressing type must be used and the first two EEPROM data structures must contain dedicated reset information.

15.4.5.2 EEPROM Calling Address

The EEPROM calling address is 0b101_0000. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. Any additional EEPROMs are addressed in sequential order.

15.4.5.3 EEPROM Data Format

The I²C module expects a particular format for data to be programmed in the EEPROM. Figure 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).

15.5 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 I2C*n*CR[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.

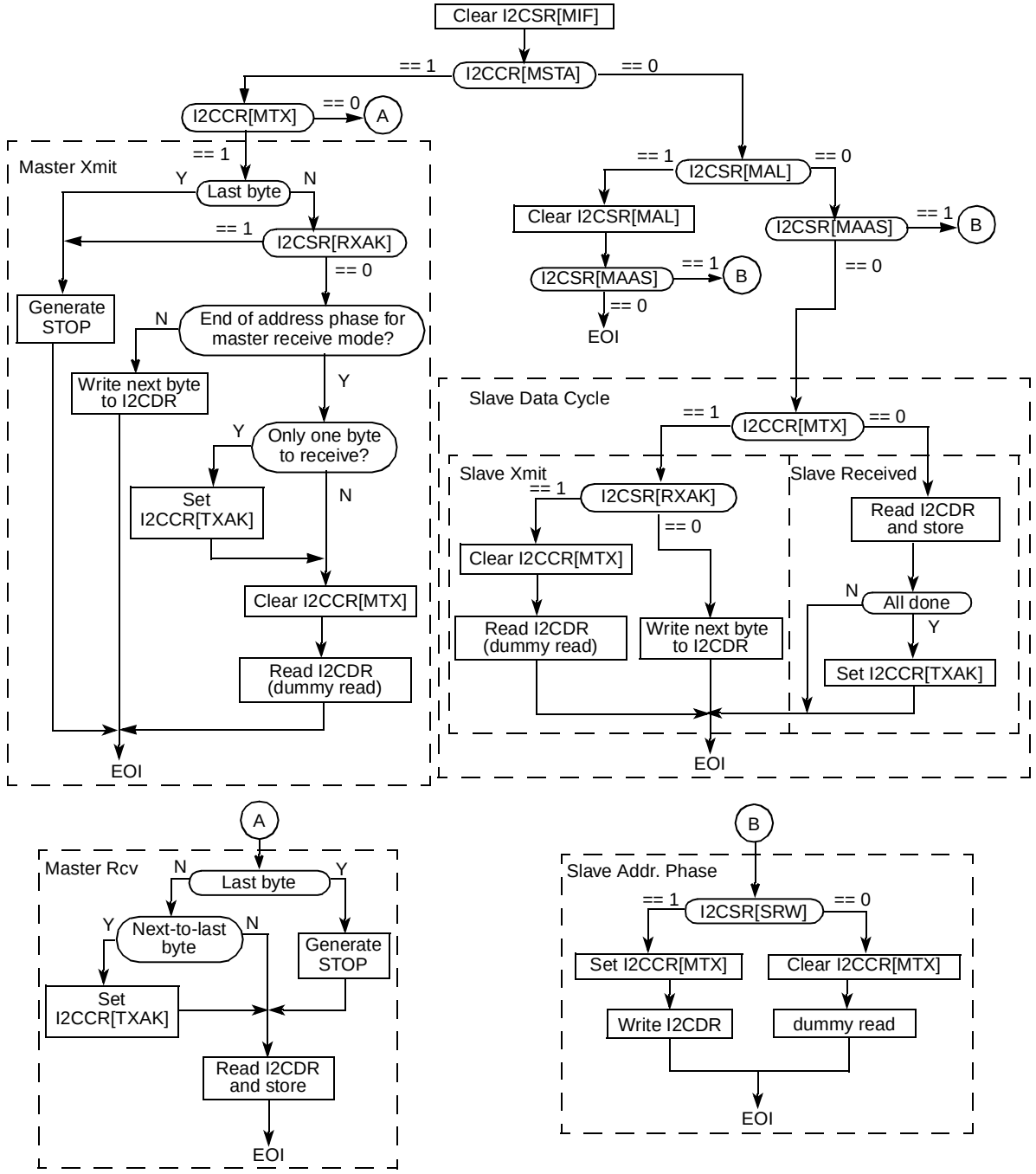


Figure 15-11. Example I²C Interrupt Service Routine Flowchart

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 I2C*n*FDR[FDR] and select the required division ratio to obtain the SCL*n* frequency from the CSB (platform) clock.
3. Update I2C*n*ADR to define the slave address for this device.
4. Modify I2C*n*CR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2C*n*CR[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 (I2C*n*SR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2C*n*CR[MSTA]) to transmit serial data and select transmit mode (set I2C*n*CR[MTX]) for the address cycle.
3. Write the slave address being called into I2C*n*DR. The data written to I2C*n*DR[0–6] comprises the slave calling address. I2C*n*CR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

The scenario above assumes that the I²C interrupt bit (I2C*n*SR[MIF]) is cleared. If MIF is set at any time, an I²C interrupt is generated (provided interrupt reporting is enabled with I2C*n*CR[MIEN] =1).

15.5.4 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2C*n*SR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2C*n*SR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2C*n*CR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2C*n*SR[MIF]
2. Read the I2C*n*DR in receive mode or write to I2C*n*DR in transmit mode. Note that this causes I2C*n*SR[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, I2C*n*CR[MTX] must be toggled at this stage (see Figure 15-11).

If the interrupt function is disabled, software can service the I2C*n*DR in the main program by monitoring I2C*n*SR[MIF]. In this case, I2C*n*SR[MIF] must be polled rather than I2C*n*SR[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 I2C*n*SR[MIF] (or any other I2C*n*SR bits), software delays may be needed to give the I²C signals sufficient time to settle.

During slave-mode address cycles (I2C*n*SR[MAAS] is set), I2C*n*SR[SRW] should be read to determine the direction of the subsequent transfer and I2C*n*CR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2C*n*SR[SRW] is not valid and I2C*n*CR[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 (I2C*n*CR[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 I2C*n*CR[TXAK] is set). For 1-byte transfers, a dummy read should be performed by the interrupt service routine (see Figure 15-11). 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 I2C*n*CR[RSTA].

15.5.7 Generation of SCL*n* when SDA*n* is Negated

It is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL*n* (even though SDA*n* may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, SDA*n* can be negated low by another I²C device while this I²C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force this I²C module to generate SCL*n* so that the device driving SDA*n* can finish its transaction:

1. Disable the I²C module and set the master bit by setting I2C*n*CR to 0x20.
2. Enable the I²C module by setting I2C*n*CR to 0xA0.
3. Read I2C*n*DR.
4. Return the I²C module to slave mode by setting I2C*n*CR 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 I2C*n*SR[MAAS] is set, software should set the transmit/receive mode select bit (I2C*n*CR[MTX]) according to the R/$\overline{W}$ command bit (I2C*n*SR[SRW]). Writing to I2C*n*CR 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 I2C*n*DR for slave transmits or dummy reading from

I2C*n*DR in slave-receive mode. The slave negates SCL*n* between byte transfers. SCL*n* is released when I2C*n*DR is accessed in the required mode.

15.5.8.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit (I2C*n*SR[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 (I2C*n*SR[RXAK] is set), the slave transmitter interrupt routine must clear I2C*n*CR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2C*n*DR then releases SCL*n* so that the master can generate a STOP condition. See Figure 15-11.

15.5.8.2 Loss of Arbitration and Forcing of Slave Mode

When a master loses arbitration the following conditions all occur:

- I2C*n*SR[MAL] is set
- I2C*n*CR[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 I2C*n*SR[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

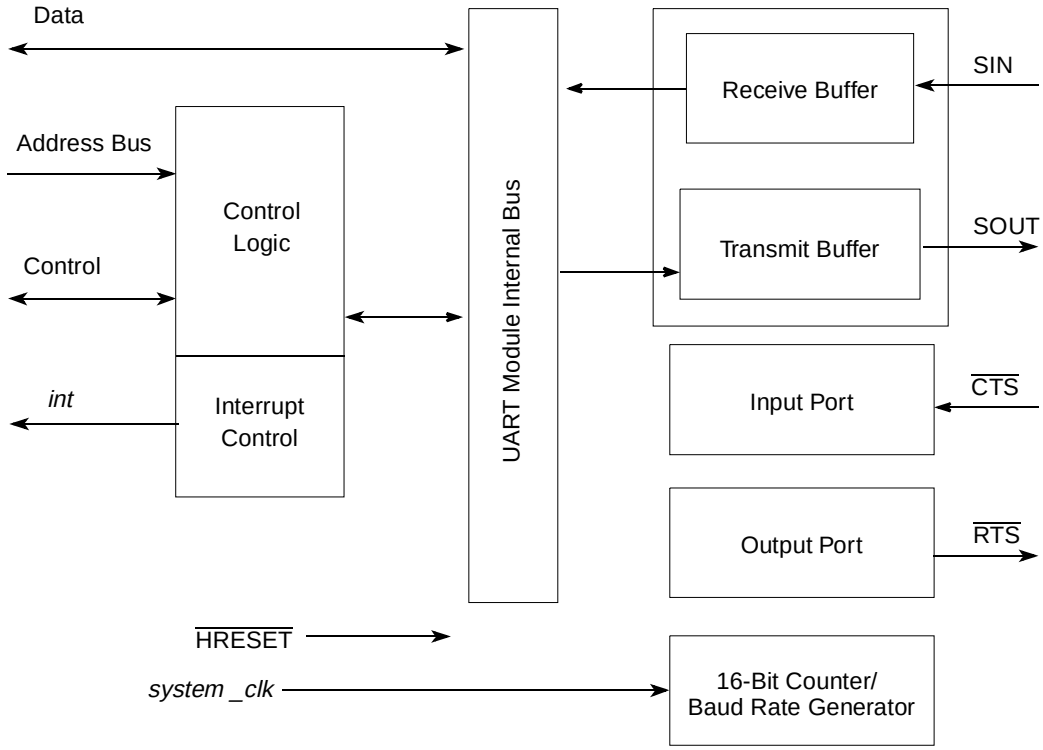


Figure 16-1. UART Block Diagram

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]	I	Serial data in. Data is received on the receivers of UART1 and UART2 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]	O	Serial data out. The serial data output signals for the UART1 and UART2 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.

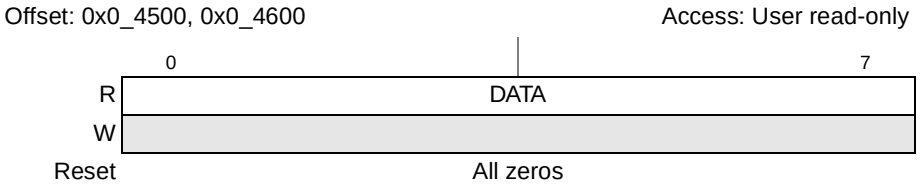


Figure 16-2. Receiver Buffer Registers (URBR1 and URBR2)

Table 16-4 describes URBR.

Table 16-4. URBR Field Description

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.

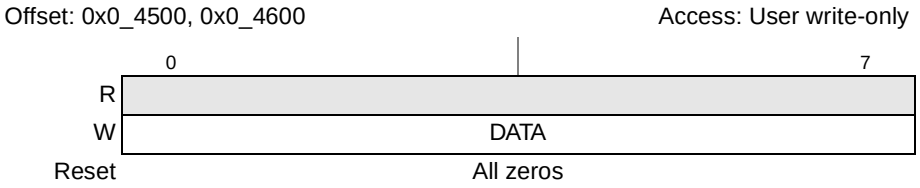


Figure 16-3. Transmitter Holding Registers (UTHR1 and UTHR2)

Table 16-5 describes the UTHR.

Table 16-5. UTHR Field Description

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 x [UDMB||UDLB]). Equivalently,

þþ [UDMB||UDLB:0b0000]þ = 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.

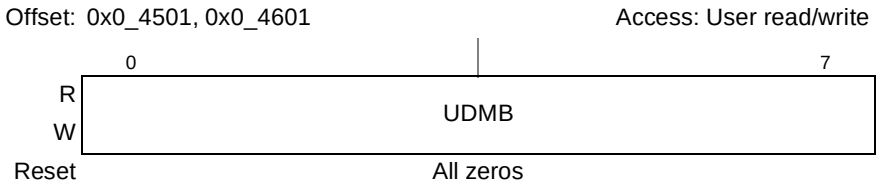


Figure 16-4. Divisor Most Significant Byte Registers (UDMB1 and UDMB2)

Table 16-6 describes the UDMB.

Table 16-6. UDMB Field Description

Bits	Name	Description
0–7	UDMB	Divisor most significant byte

Figure 16-5 shows the bits in the UDLBs.

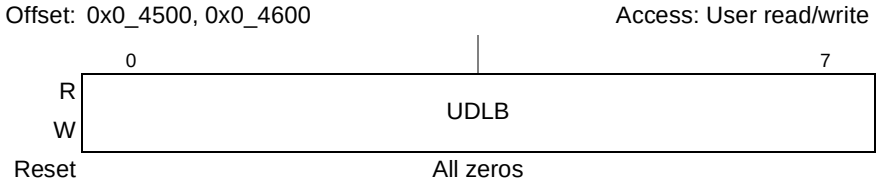


Figure 16-5. Divisor Least Significant Byte Registers (UDLB1 and UDLB2)

Table 16-7 describes the UDLB.

Table 16-7. UDLB Field Description

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.

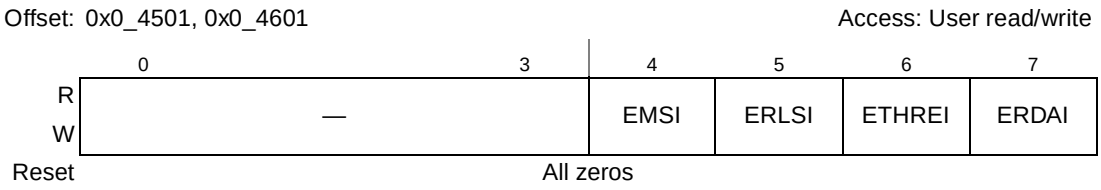


Figure 16-6. Interrupt Enable Registers (UIER1 and UIER2)

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.

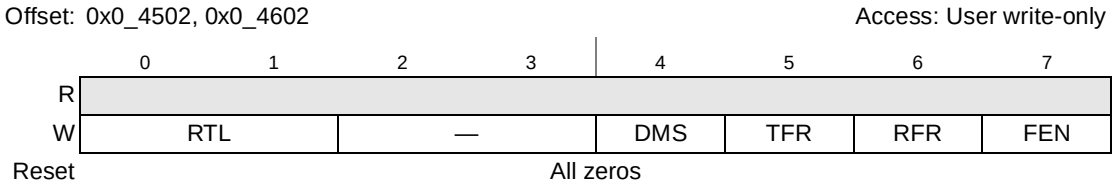


Figure 16-8. FIFO Control Registers (UFCR1 and UFCR2)

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.

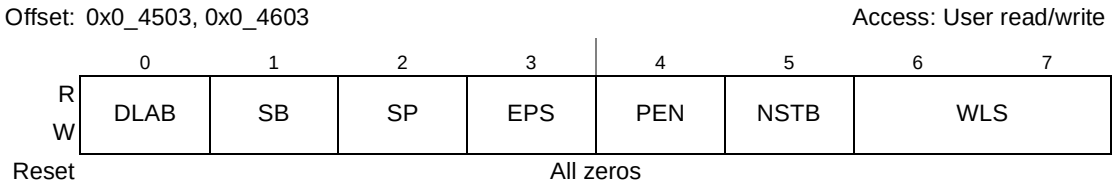


Figure 16-9. Line Control Register (ULCR1 and ULCR2)

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.

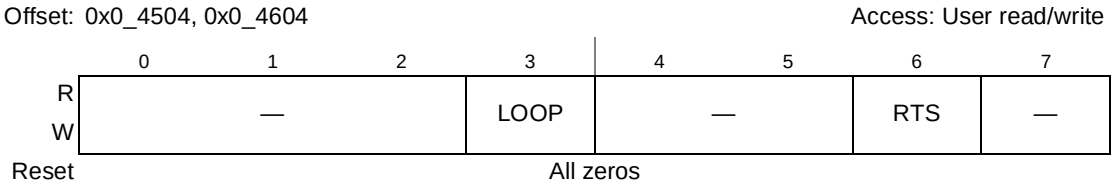


Figure 16-10. Modem Control Register (UMCR1 and UMCR2)

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.

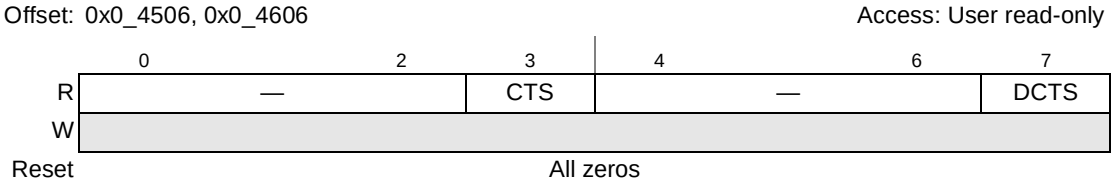


Figure 16-12. Modem Status Register (UMSR1 and UMSR2)

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.

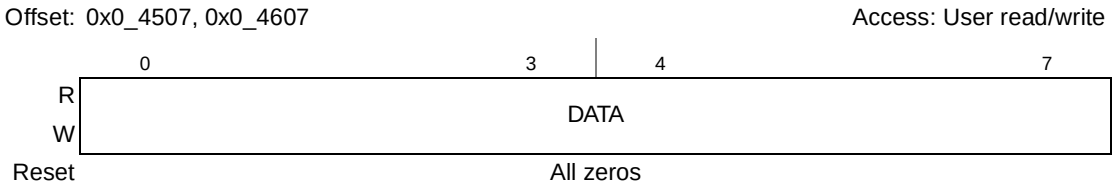


Figure 16-13. Scratch Register (USCR)

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.

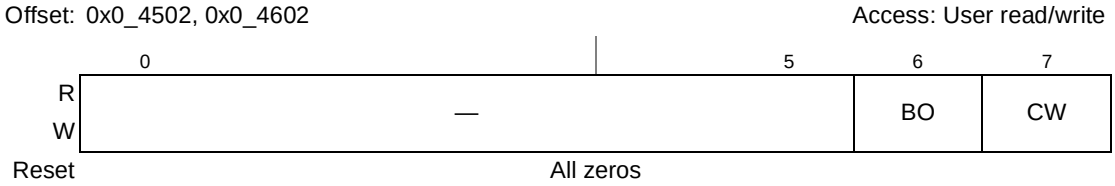


Figure 16-14. Alternate Function Register (UAFR)

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.

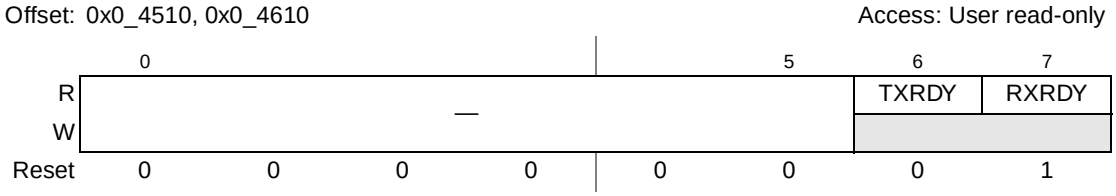


Figure 16-15. DMA Status Register (UDSR)

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

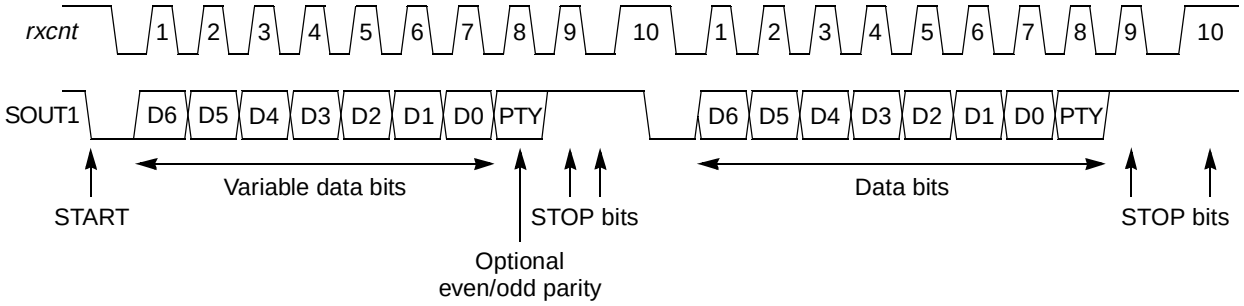
The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the system clock signal.

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.

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) x (system clock frequency/divisor value)

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling 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.

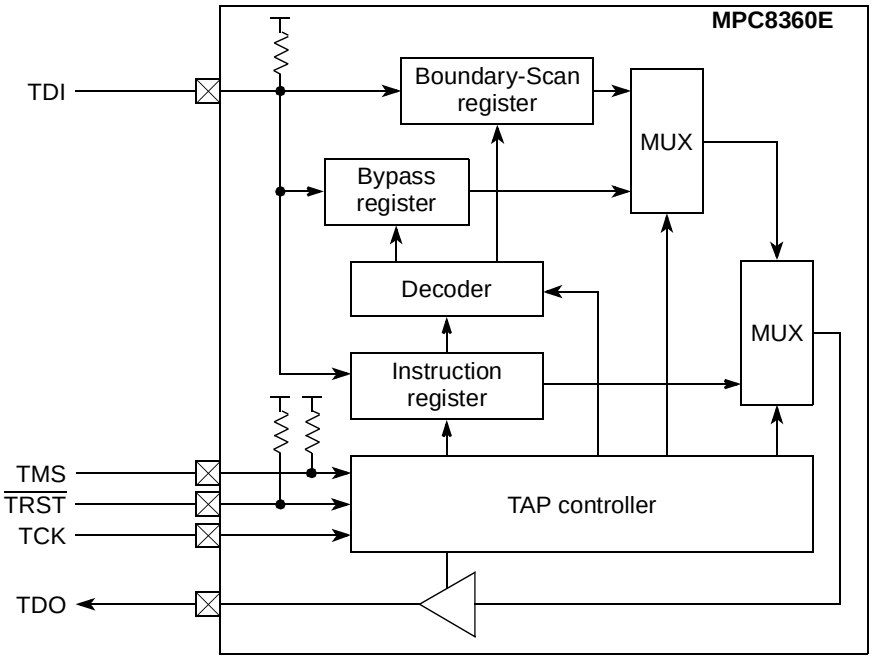


Figure 17-1. JTAG Interface Block Diagram

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 duty cycle as specified in the MPC8360E Hardware Specification, Rev 0. 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 standard for more details.

Table 17-2. JTAG Test—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDI	I	JTAG test data input.	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
TDO	O	JTAG test data output.	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 standard for more details.
TMS	I	JTAG test mode select.	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
$\overline{\text{TRST}}$	I	JTAG test reset.	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the 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 standard for more details.

17.3 JTAG Registers and Scan Chains

The bypass, boundary-scan, and instruction JTAG registers and their associated scan chains are mandatory for compliance with the IEEE 1149.1 specification.

- Bypass register. The bypass register is a single-stage register used to bypass the boundary-scan latches of the device during board-level boundary-scan operations involving components other than the device. The use of the bypass register reduces the total scan string size of the boundary-scan test.
- Boundary-scan registers. The JTAG interface provides a chain of registers dedicated to boundary-scan operations. To be JTAG-compliant, these registers cannot be shared with any functional registers of the device. The boundary-scan register chain includes registers controlling the direction of the input/output drivers, in addition to the registers reflecting the signal value received or driven.

  The boundary-scan registers capture the input or output state of the device's signals during a Capture_DR TAP controller state. When a data scan is initiated following the Capture_DR state, the sampled values are shifted out through the TDO output while new boundary-scan register values are shifted in through the TDI input. At the end of the data scan operation, the

boundary-scan registers are updated with the new values during an update_DR TAP controller state.

- Instruction register. The 8-bit JTAG instruction register serves as an instruction and status register. As TAP controller instructions are scanned in through the TDI input, the TAP controller status bits are scanned out through the TDO output.
- TAP controller. The device provides a standard JTAG TAP controller that controls instruction and data scan operations. The TMS signal controls the state transitions of the TAP controller.

Chapter 18
Delay Lock Loop (DLL)

This chapter describes the theory of operation of the delay lock loop (DLL) module in the integrated device. Additionally, the configuration, control, and status registers are described. Note that other chapters in this book describe additional specific initialization aspects for individual blocks.

18.1 Introduction

The DLL unit consists of a phase detection circuit, adjustable delay unit, register file and control logic. A high-level block diagram of these elements is shown in Figure 18-1.

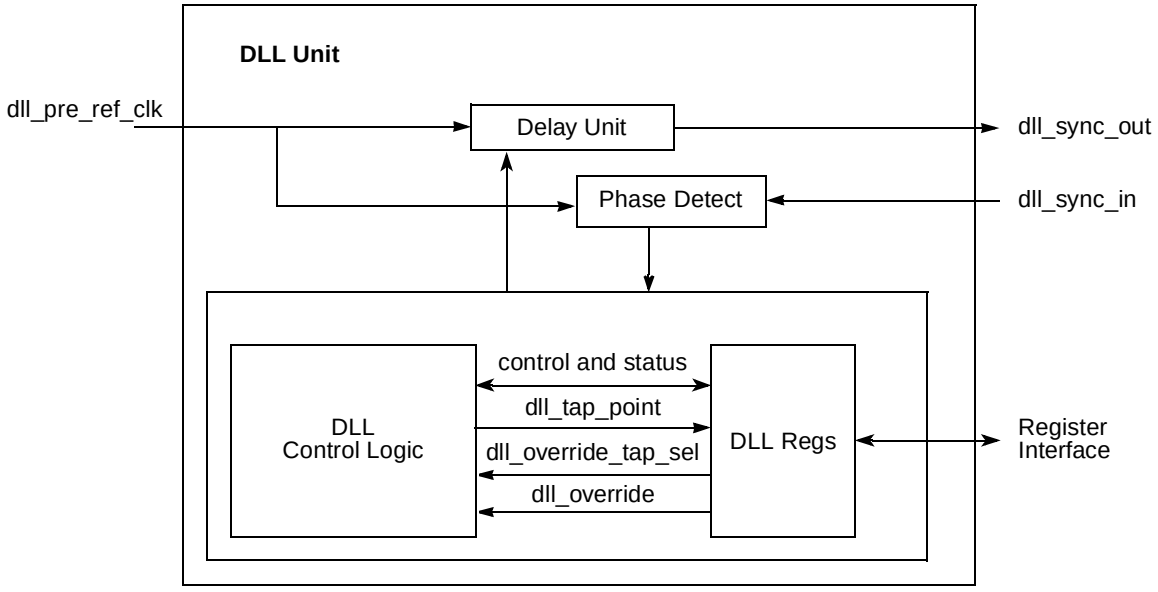


Figure 18-1. DLL Block Diagram

18.2 Overview

The purpose of the DLL macro is to shift a reference clock to create an output clock to be used by external memory or I/O devices. The output clock is created by propagating the reference clock through a delay chain. A phase detect circuit compares the phase of the reference clock with that of a feedback clock. An output TAP point of the delay chain is chosen for the output clock such that the feedback clock is phase aligned with the reference clock. The DLL macro also contains override, debug, and error features.

DLL is used for LCLK (local bus clocks) skew elimination. Therefore, DLL_SYNC_OUT/DLL_SYNC_IN should be interpreted as LSYNC_OUT/LSYNC_IN, and DLL_CLK_OUT[0:n] should be interpreted as LCLK[0:n].

18.2.1 Features

- LOCK status reports when phase lock search was completed.
- WRAP status reports when the DLL wraps beyond either end of its delay chain during the search process. If WRAP is set while LOCK is set, this indicates a failure to find a lock during the search process.
- OVERRIDE mode allows a specific tap point to be selected.
- Current tap point is continuously reported.

18.2.2 Modes of Operation

- Normal mode: The DLL can operate in normal mode, in which the phase detector circuitry dynamically adjusts the tap point of the delay line.
- Override mode: The DLL can operate in override mode in which the tap point is specified by an input vector.

18.2.3 External Signals

Table 18-1 displays the signals of the DLL macro.

Table 18-1. DLL Macro External Signals

Signal	I/O	Signal Description
DLL_SYNC_OUT	O	DLL output clock. Phase aligned with other clock outputs (DLL_CLK_OUT[0:n]).
DLL_SYNC_IN	O	DLL feedback clock, phased aligned with the internal logic clock.
DLL_CLK_OUT[0:n]	O	DLL output clocks to be used by external on-board devices.

18.3 Initialization and Application Information

An example application using the DLL macro is shown in Figure 18-2. As this example shows, a memory or I/O controller generates the reference clock based on an internal knowledge of the ratio between the external clock and internal clk. DLL_CLK_OUT goes to the external memory or I/O devices, and the DLL_SYNC_OUT must be connected back to the DLL_SYNC_IN while keeping the same trace length as of DLL_CLK_OUT path. The DLL macro uses the phase detection and the delay chain to align the internal logic clock with the external device input clock, such that the integrated device's flip-flops and the external device's flip-flops are fed by clocks with minimal skew between them.

DLL lock status is used by the memory controller to hold off memory access until DLL search has been completed. However, if DLL override mode is used, DLL lock has no meaning.

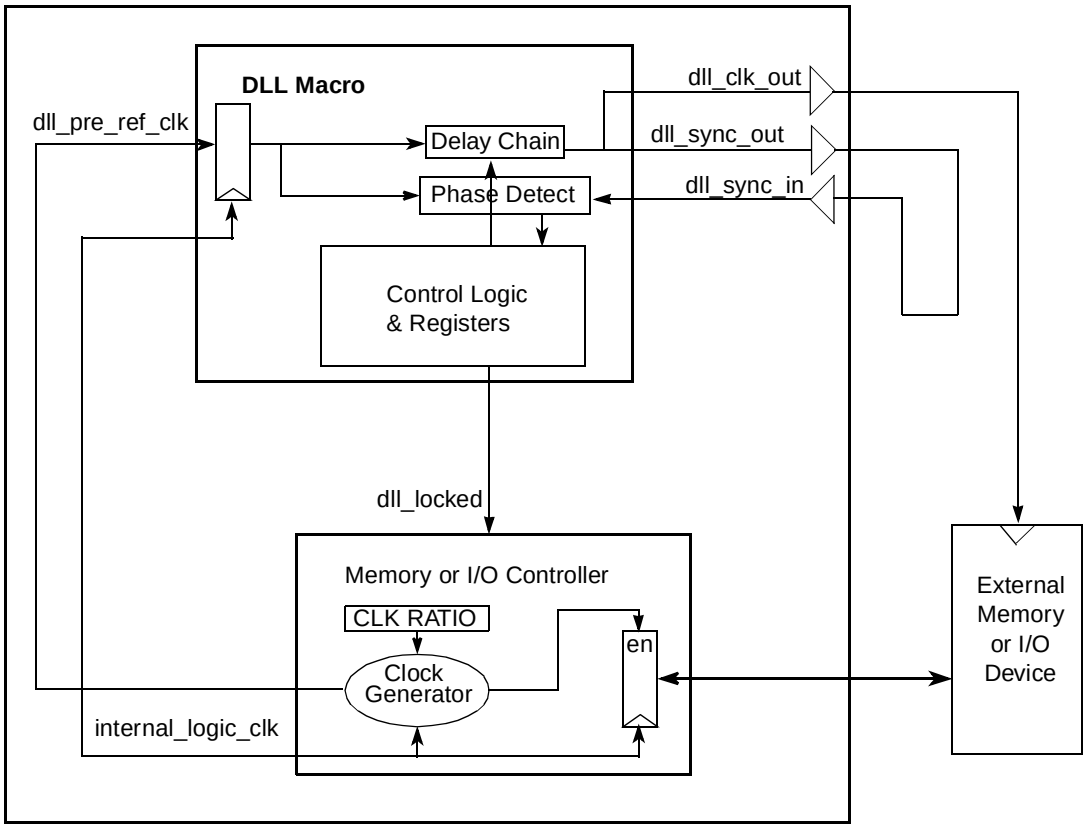


Figure 18-2. DLL Application Example

18.4 Memory Map/Register Definition

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

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

Table 18-2 shows the DLL registers.

Table 18-2. DLL Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Reserved. Reset value should be preserved.	R/W	0x0500_0280	—
0x04	Reserved. Reset value should be preserved.	R/W	0x8004_0810	—
0x08	DLL override register (DLLOVR)	R/W	0x0000_0000	18.4.1/18-4
0x0C	DLL status register (DLLSR)	R	0x0000_0000	18.4.2/18-4
0x10	DLL clock register (DLLCK)	R/W	0xFC00_0000	18.4.3/18-5
0x14–0xFF	Reserved.	—	—	—

18.4.1 DLL Override Register (DLLOVR)

The DLL override register (DLLOVR), shown in Figure 18-3, controls the override operation and the override value (coarse or fine) to force on the DLL in override mode.

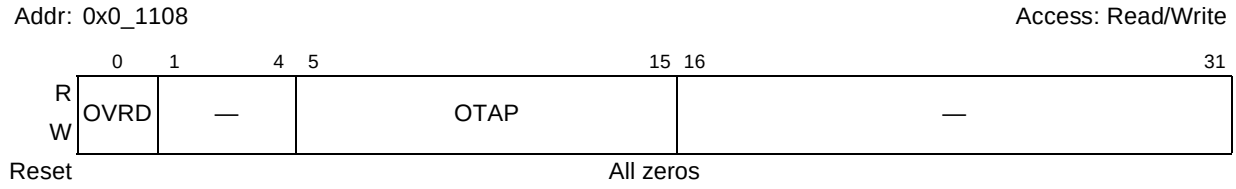


Figure 18-3. DLL Override Register

DLLOVR can be used to force a fixed delay value. It is recommended to wait until DLL lock is achieved, read the measured course and TAP delay values from the DLL status register, and then set the override delay values accordingly. Table 18-3 describes DLLOVR fields.

Table 18-3. DLLOVR Field Description

Bits	Name	Description
0	OVRD	DLL override mode. 0 Normal mode 1 Override mode
1–4	—	Reserved
5–15	OTAP	DLL override coarse and TAP select. OTAP[5–7] Coarse delay select OTAP[8–15] TAP delay select
16–31	—	Reserved

18.4.2 DLL Status Register (DLLSR)

DLLSR shown in Figure 18-4, indicates the DLL status. It is a read only register.

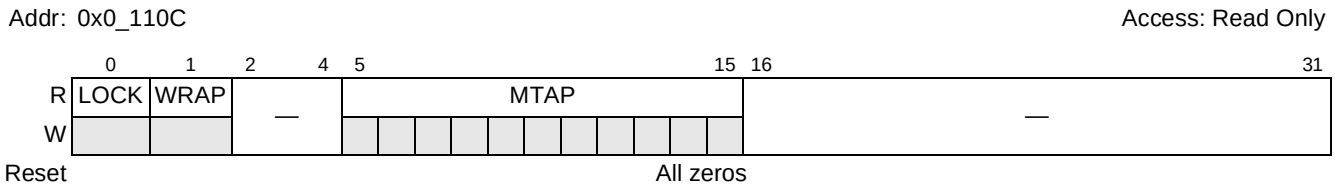


Figure 18-4. DLL Status Register

Table 18-4 describes DLLSR fields.

Table 18-4. DLLSR Field Description

Bits	Name	Description
0	LOCK	DLL locked. When the DLL finished its TAP point search, the LOCK bit is set. The LOCK bit must be considered together with the WRAP bit state.
1	WRAP	DLL wrapped. When the DLL LOCK is set, WRAP provides information for the DLL condition. If WRAP = 1, it indicates that the DLL search for a TAP point has completed unsuccessfully. When the DLL search was successful, the WRAP state remains clear while LOCK is set.
2–4	—	Reserved
5–15	MTAP	Measured coarse and tap delay by the DLL. MTAP[5–7] Coarse delay MTAP[8–15] Tap delay
16–31	—	Reserved

18.4.3 DLL Clock Register (DLLCK)

DLLCK shown in Figure 18-5, enables or disables the signals clock out.

Addr: 0x0_1110 Access: Read/Write

	0	1	2	3																										31
R W	CE0	CE1	CE2	—																										
Reset	1	1	1	1	1	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

Figure 18-5. DLL Clock Register

Table 18-5 describes DLLCK fields.

Table 18-5. DLLCK Field Description

Bits	Name	Description
0	CE0	Enable/Disable LCLK[0] signal clock out 0 Disable LCLK[0] 1 Enable LCLK[0]
1	CE1	Enable/Disable LCLK[1] signal clock out 0 Disable LCLK[1] 1 Enable LCLK[1]
2	CE2	Enable/Disable LCLK[2] signal clock out 0 Disable LCLK[2] 1 Enable LCLK[2]
3–31	—	Reserved

Part IV
QUICC Engine Block

Part IV describes the QUICC Engine block of the MPC8360 integrated processor. The following chapters are included:

- Chapter 19, “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 20, “QUICC Engine Block Control,” 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. The QUICC Engine control registers 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 21, “QUICC Engine 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 (UCC’s, UPC’s, TDM’s, etc.).
- Chapter 22, “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 23, “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 24, “UCC as Slow Communications Controllers,” 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 25, “UCC 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 26, “UCC 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 27, "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, high-speed serials (Ethernet, HDLC, HDLC bus, Transparent), and Ethernet for the first Mile (EFM). Chapter 23, "Unified Communications Controllers (UCCs)," should be read before reading this chapter.
- 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, "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 30, "UCC Ethernet Controller (UEC)," describes the Ethernet IEEE 802.3 protocol, which is a widely-used LAN based on the carrier-sense multiple access/collision detect (CSMA/CD) approach.
- Chapter 31, "QUICC Engine IEEE1588 Assist," covers the IEEE1588 implementation, which, in the QUICC Engine module, is a hardware-assisted method. The hardware assist includes a time stamp unit to recognize PTP frames and transfer relevant timestamps, and a real time clock with high resolution.
- Chapter 32, "ATM Controller AAL0, AAL1, 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, AAL1 circuit emulation service (CES), AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.
- Chapter 33, "UTOPIA POS Bus Controller (UPC)," covers the UPC (UTOPIA/POS-PHY L2 bus controller), which is the UTOPIA/POS-PHY MAC peripheral of the QUICC Engine block. The QUICC Engine block supports UTOPIA/POS-PHY level 2 for both master and slave modes.
- Chapter 34, "Multi-Channel Controller (MCC)," addresses the multi-channel controller (MCC), which supports up to 256 separate time-division serial channels. The MCC is paired with a serial interface (SI), allowing it to communicate over any of the SI's 8 time-division multiplexed streams (TDM). Proper programming of the SI and SIRAM is necessary for routing the timeslots within a TDM stream to the appropriate MCC channel at the desired time. Users should be familiar with programming the SI and parallel I/O ports before proceeding.
- Chapter 35, "ATM AAL1 Circuit Emulation Service," describes implementation of circuit emulation service (CES) using ATM adaptation layer type 1 (AAL1) on the QUICC Engine module and should be used as a supplement to Chapter 32, "ATM Controller AAL0, AAL1, and AAL5."

- Chapter 36, “Serial Interface with Time-Slot Assigner,” pertains to the serial interface (SI), which manages the routing of eight TDM lines to the QUICC Engine block serial drivers, the MCC and 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 37, “Point-to-Point Protocol (PPP),” describes the QUICC Engine implementation of the point-to-point protocol (PPP), which is compliant with the RFCs 1661, 1662, 1990, 2686, 3153. This chapter covers the functionality and data structures of the PPP protocol, the Multilink-MultiClass PPP (ML-MC) protocol, and the PPP mux protocol.
- Chapter 38, “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 39, “Enhanced MSP Microcode,” describes the QUICC Engine multi service platform derivative (QUICC Engine-EMSP), which adds ATM layer functionality, suitable for the implementation of various network applications, such as a port controller on subscriber line cards, ATM cell processing, cable modem controllers, ATM multiplexing and concentrating systems and other multiservice and multiprotocol applications especially in the branch/enterprise dial access applications.
- Chapter 40, “L2 Ethernet Switch,” describes the L2 Ethernet switch in the QUICC Engine block. This switch offers up to eight connection ports of 10/100 Mbps with MII/RMII Ethernet external ports and one CPU internal port. It also provides VLAN functionality, IGMP snooping, store-and-forward switching operation, and packet-error filtering.
- Chapter 41, “E2AAL2 Microcode,” addresses the implementation of the ATM adaptation layer type 2 (AAL2), compliant with the ITU-T recommendations I.363.2 and I.366.1. This chapter describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR, and is meant to be used as a supplement to Chapter 32, “ATM Controller AAL0, AAL1, and AAL5.”
- Chapter 42, “QMC (QUICC Multi-Channel Controller),” 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 43, “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 44, “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.

Chapter 19
System Interface

The QUICC Engine system interface consists of functions that interface to the coherent system bus and CPU. The system interface includes the following:

- Serial DMA
- Interrupt Controller

The first interface to the system is the serial DMA. This 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 CPU to verify the cause of an interrupt from the QUICC Engine module.

19.1 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 implements two dedicated virtual SDMA channels for each peripheral (e.g. UCC, MCC, SPI), one for the receiver and one for the transmitter. Additional virtual channels are available for general purpose accesses to external memory.

19.1.1 Data Paths

Figure 19-1 is a simplified block diagram that shows the data paths. They may be configured with one DDR bus (32 or 64 bit data) or with two DDR busses (32-bit data). As shown in the figures, depending on the external bus configuration, the secondary bus of the QUICC Engine module may be connected to the second DDR and/or to the Local bus. Accesses of the QUICC Engine module to the secondary bus are not seen on the coherent system bus. This allows for concurrent bus transactions on the two buses (marked as ‘1’ and ‘2’ in the figure).

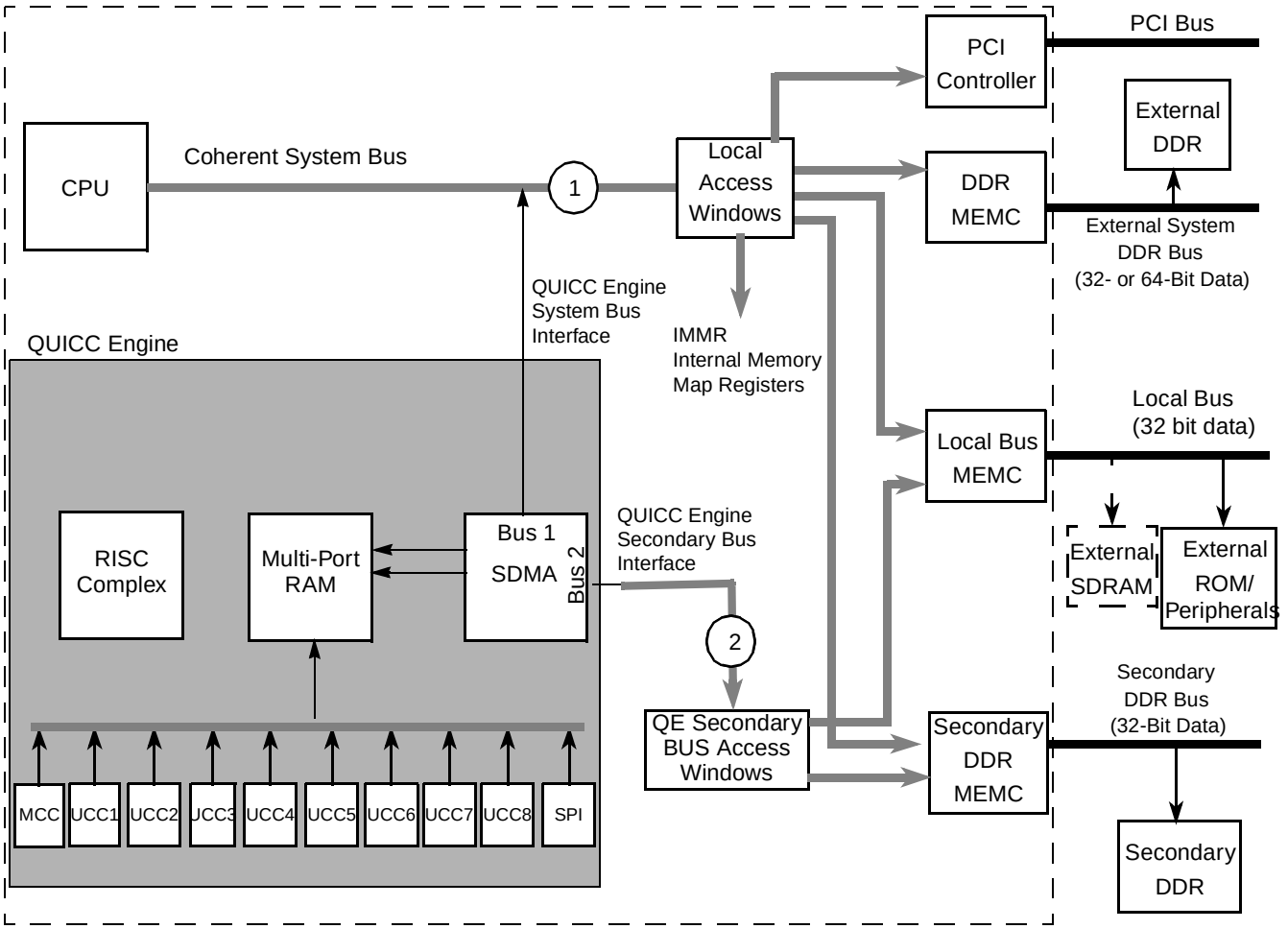


Figure 19-1. Data Paths

The ‘Local Access Windows’ block shown in the figure, is responsible for distribution of coherent system bus transactions to the various possible targets (e.g. 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 or the Secondary DDR 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, MCC, SPI). Refer to each protocol of these peripherals for programming of this feature.

19.1.2 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_1,2_MSK]) in the SDMA status register (SDSR),

2. The CPU in its interrupt service routine reads the SDSR to determine which bus generated the error (System bus or secondary bus),
3. The system recovery from bus error is dependent upon how the user configured the SBER_1,2 bits in the SDMR register. 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 (reset command to the QUICC Engine Command Register is not sufficient).

In general, it should be noted that the CPU, 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 20.7, “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.

19.1.2.1 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 reduce the complexity of the recovery flow.

19.1.2.2 Selective Peripheral Recovery Procedure

Selective recovery can be a complex procedure due to the following:

- A peripheral (UCC or MCC) can have dependencies with other peripherals (as a part of an interworking controller like L2 switch, ATM switch, ML-PPP 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:

1. The SNUM is of a peripheral which implement one of the following termination protocols:
   — HDLC
   — Transparent
   — UART
   — BiSync
   — QMC
   — MCC (HDLC, Transparent or SS#7 channels only)
   — SPI
   — USB

In other cases it is recommended to reset the QUICC Engine module and do a full initialization sequence.

19.1.3 SDMA and Reset

When the QUICC Engine is reset, through the RST bit in CECR - See Section 20.3.1, "QUICC Engine Command Register (CECR)," the SDMA continues to process outstanding transactions in it's FIFOs although the data related to these transactions may be corrupted. During system reset ($\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$) all SDMA FIFOs are flushed thus all outstanding transactions are killed.

19.1.4 Bus Arbitration

19.1.4.1 Arbitration Over the System Bus

The QUICC Engine module arbitrates over the system bus by requesting access tothe 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 system arbiter at two possible priority levels. When the SDMA is in normal state it requests the bus 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 the bus 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).

19.1.4.2 Arbitration Over the Secondary Bus

Arbitration over the Secondary bus is not affected by internal state of the SDMA. Generally arbitration occurs in two places (see Figure 19-1): 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 it's secondary bus, and another master which initiated a transaction on the System Bus. In the case of the Local Bus Memory Controller, the QUICC Engine module has the lower priority. In the case of the Secondary DDR SDRAM Controller arbitration is based on rotating priority.

19.1.4.3 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[ER1,2_MASK].

19.1.5 SDMA and Snooping

The SDMA in the QUICC Engine module is able to dynamically signal to the CPU 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 CPU. Refer to each protocol of these peripherals for programming of this feature, for example, for Ethernet, refer to Section 30.5.4, "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 CPU.

19.1.6 Bus Selection Mechanisms

For each transaction initiated by the RISC complex, 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, e.g. for Ethernet, refer to Section 30.5.4, "Bus Mode Register (BMRx)."

The selection between these two modes is made in SDMR[ADR_SEL].

19.1.7 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 3Kbytes in the multi-user RAM. Refer for the detailed CEN bits field description for suggested values.

19.1.8 Programming Model of the SDMA

All of the DMA registers are accessible via software (through the slave) for read/write.

During hreset and sreset, 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.

19.1.8.1 Serial DMA Status Register (SDSR)

SDSR, shown in Figure 19-2, 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.

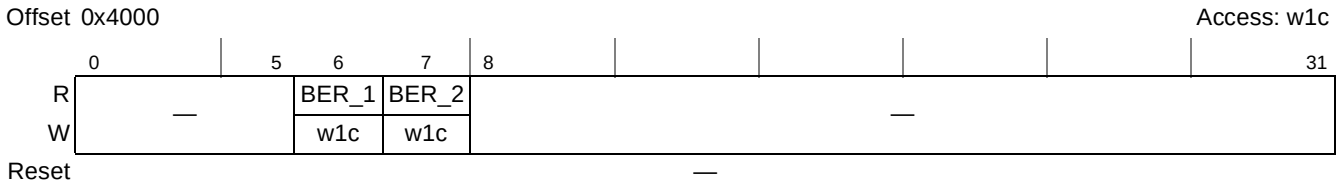


Figure 19-2. Serial DMA Status Register (SDSR)

Table 19-1 describes the SDSR fields.

Table 19-1. 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

Table 19-1. SDSR Field Descriptions (continued)

Bits	Name	Description
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.

19.1.8.2 Serial DMA Mode Register (SDMR)

SDMR, shown in Figure 19-3, 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

	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

	8	9	11	12	13	14	15
R W	EB1_MSK	—		ER1_MSK	ER2_MSK	—	
Reset	0	0	0 0	0	0	0	0

	16	18	19	21	22	23
R W	CEN		—		SBER_1	SBER_2
Reset	1	0 1	0	0 0	0	0

	24	25	26	27	28	29	31
R W	EB1_PR		—		ER1_PR	—	
Reset	0	0	0	0	0	0 0	0

Figure 19-3. Serial DMA Mode Register (SDMR)

Table 19-2 describes the SDMR fields.

Table 19-2. 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	—	Reserved, should be cleared.
10–11	—	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 - 1Kbytes 010 - 1.5Kbytes 011 - 2KBytes 100 - 2.5Kbytes 101 - 3KBytes 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 19-2. 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.

19.1.8.3 Serial DMA Threshold Registers (SDTR1 and SDTR2)

SDTR1 and SDTR2, shown in Figure 19-7, 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 dmar 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).

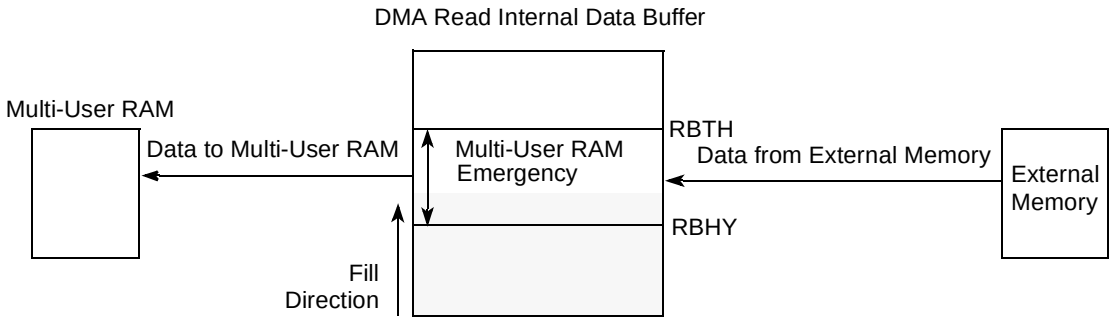


Figure 19-4. DMA Read Data Path

When the internal DMA Write data buffer (buffer for dmaw 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).

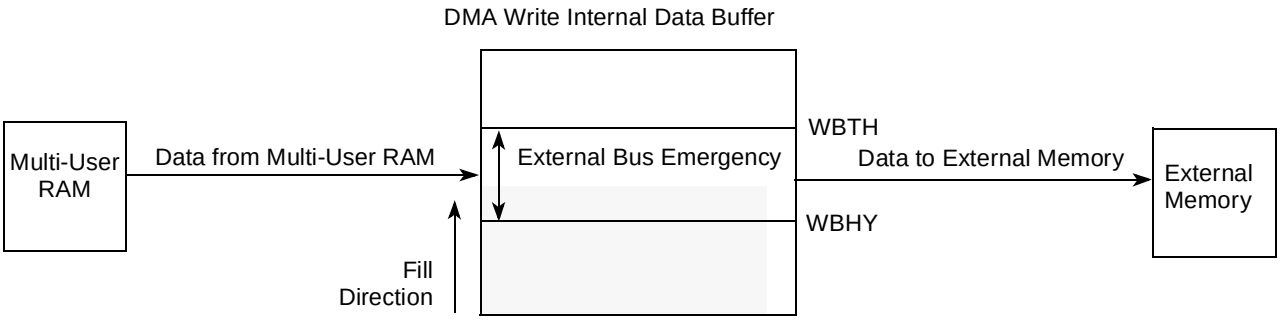


Figure 19-5. DMA Write Data Path

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 & External bus will be asserted, and will be held asserted until the command queue reaches the hysteresis value (CQHY).

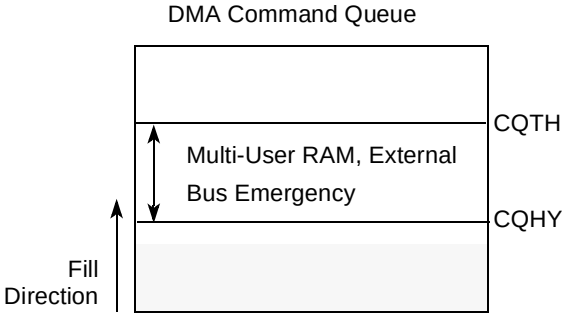


Figure 19-6. DMA Command Queue

Offset 0x4008
0x400C

Access: Read/Write

	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 19-7. Serial DMA Threshold Registers (SDTR1 and SDTR2)

Table 19-3 describes the SDTRx fields.

Table 19-3. 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 19-3. 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

19.1.8.4 Serial DMA Hysteresis Registers (SDHY1 and SDHY2)

SDHY1 and SDHY2, shown in Figure 19-8, 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 SDTRx registers. The high priority indicator will be held until the parameter reaches the hysteresis value, found in the SDHYx registers. For a full explanation, see Section 19.1.8.3, “Serial DMA Threshold Registers (SDTR1 and SDTR2),” on page 19-9.

Offset 0x4010
0x4014

Access: Read/Write

	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 1 0	0 0 0 0

Figure 19-8. Serial DMA Hysteresis Registers (SDHY1 and SDHY2)

Table 19-4 describes the SDHYx fields.

Table 19-4. 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 19-4. 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).

19.1.8.5 Serial DMA Transfer Address Registers (SDTA1 and SDTA2)

SDTA1 and SDTA2, shown in Figure 19-9, 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.

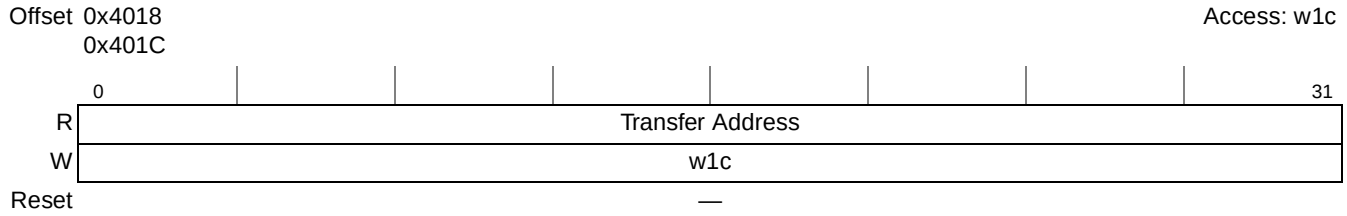


Figure 19-9. Serial DMA Transfer Address Registers (SDTA1 and SDTA2)

Table 19-5 describes the SDTAx fields.

Table 19-5. 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).

19.1.8.6 Serial DMA Transfer Communication Channel Number (MSNUM) Registers (SDTM1 and SDTM2)

SDTM1 and SDTM2, shown in Figure 19-10, 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.

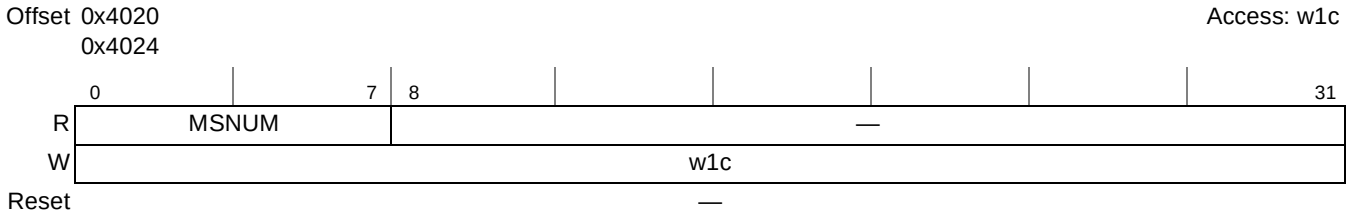


Figure 19-10. Serial DMA Transfer MSNUM Registers (SDTM1 and SDTM2)

Table 19-6 describes the SDTM*x* fields.

Table 19-6. 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.

19.1.8.7 Serial DMA Address Qualify Registers (SDAQR)

SDAQR, shown in Figure 19-11, 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.

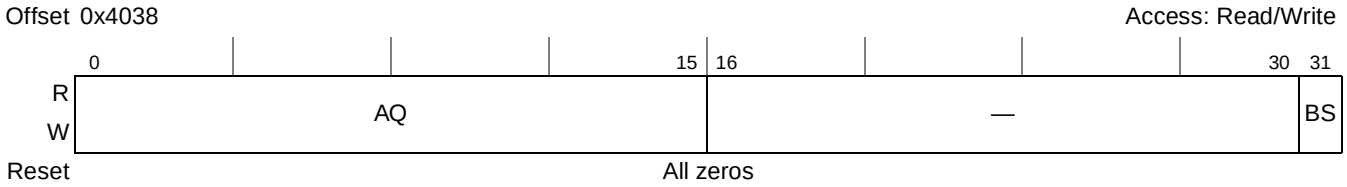


Figure 19-11. Serial DMA Address Qualify Register (SDAQR)

Table 19-7 describes the SDAQR fields.

Table 19-7. SDAQR Field Descriptions

Bits	Name	Description
0–15	AQ	16 msbs of the address match.
16–30	—	Reserved, should be cleared.
31	BS	Bus select in case of address match 1- bus1(if match), bus2 (if no match) 0- bus2 (if match), bus1 (if no match)

19.1.8.8 Serial DMA Address Qualify Mask Register (SDAQMR)

SDAQMR, shown in Figure 19-12, 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.

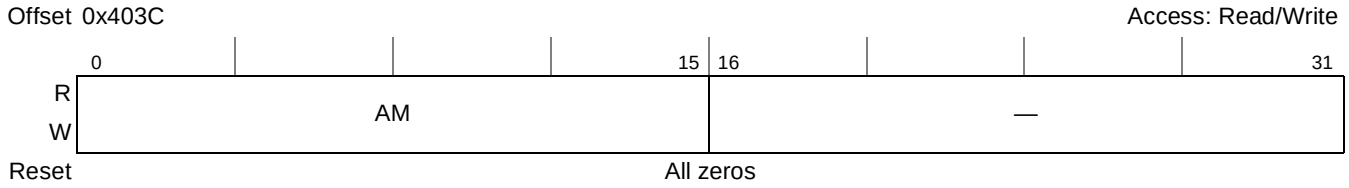


Figure 19-12. Serial DMA Address Qualify Mask Register (SDAQMR)

Table 19-8 describes the SDAQMR fields.

Table 19-8. 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.

19.1.8.9 Serial DMA Temporary Buffer Base in Multi-User RAM Value (SDEBCR)

SDEBCR, shown in Figure 19-13, holds the Temporary Buffer base address in multi-user RAM.

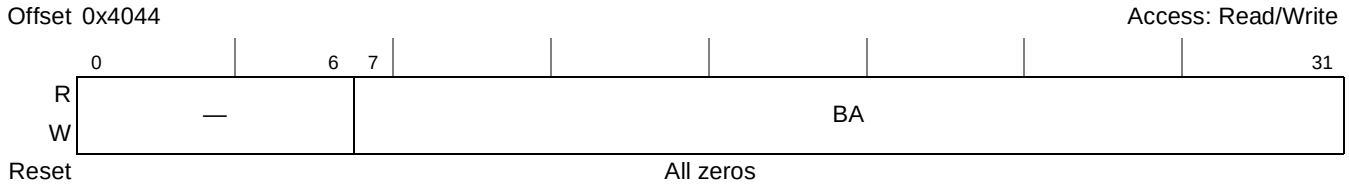


Figure 19-13. Serial DMA TEMPORARY BUFFER Base in Multi-User RAM Value (SDEBCR)

Table 19-9 describes the SDEBCR fields.

Table 19-9. 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.

19.2 Interrupt Controller

The peripherals of the QUICC Engine module (UCCs, USB, MCC, SDMA, Timers, and SPIs) 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.

19.2.1 Interrupt Configuration

Figure 19-14 shows the QUICC Engine interrupt structure. The interrupt controller receives interrupts from various internal sources within the QUICC Engine module.

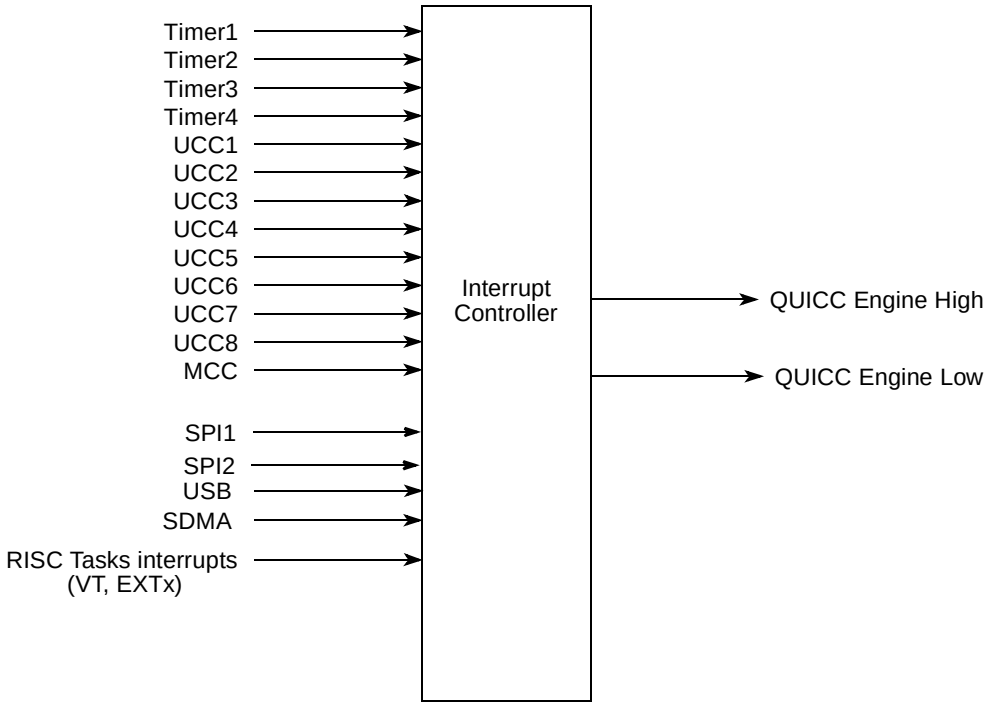


Figure 19-14. QUICC Engine Module Interrupt Structure

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, MCC, and SPIs. See Section 19.2.3, “UCC Relative Priority,” for more information.

- Assigning highest priority to one interrupt source. See Section 19.2.4, “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.

19.2.2 Interrupt Source Priorities

The interrupt controller has 18 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 19.3.2, "QUICC Engine System Interrupt Control Register (CICNR)"). All the sources can be allocated to the QUICC Engine Low interrupt output. Table 19-10 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 & YCC entries in Table 19-10, are described in Section 19.2.3, "UCC Relative Priority."

Table 19-10. 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
20	RTB2 (Spread)	Yes
21	YCC1 (Grouped)	Yes
22	YCC2 (Grouped)	Yes
23	YCC3 (Grouped)	Yes
24	YCC4 (Grouped)	Yes

Table 19-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
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
53	MIXA5 (Spread)	Yes
54	ZCC5 (Grouped)	Yes
55	ZCC6 (Grouped)	Yes
56	ZCC7 (Grouped)	Yes
57	ZCC8 (Grouped)	Yes

Table 19-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
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
89	Reserved	—
90	WCC4 (Spread)	Yes
91	ZCC4 (Spread)	Yes
92–93	Reserved	—
94	RTB7 (Spread)	Yes

Table 19-10. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
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 19.2.3, “UCC Relative Priority,” for more information.
- The highest priority option. See Section 19.2.4, “Highest Priority Interrupt,” for more information.

19.2.3 UCC Relative Priority

The relative priority between the eight UCCs is programmable and can be changed dynamically. There is no entry for UCC1-4 and UCC5-8 in Table 19-10, but rather there are entries for XCC1–XCC8 and YCC1–YCC8. UCC5-8 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 & 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 19-11.

Table 19-11. Mapping of FCCs and SCCs to UCCs for 82xx/85xx Compatibility

82xx Device	QUICC Engine Device
FCC1	UCC1
FCC2	UCC2
FCC3	UCC3
SCC1	UCC5
SCC2	UCC6
SCC3	UCC7
SCC4	UCC8

19.2.4 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.

19.2.5 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 19-10 shows the interrupt sources that have multiple interrupting events. Figure 19-15 shows an example on how the masking occurs, using a UCC.

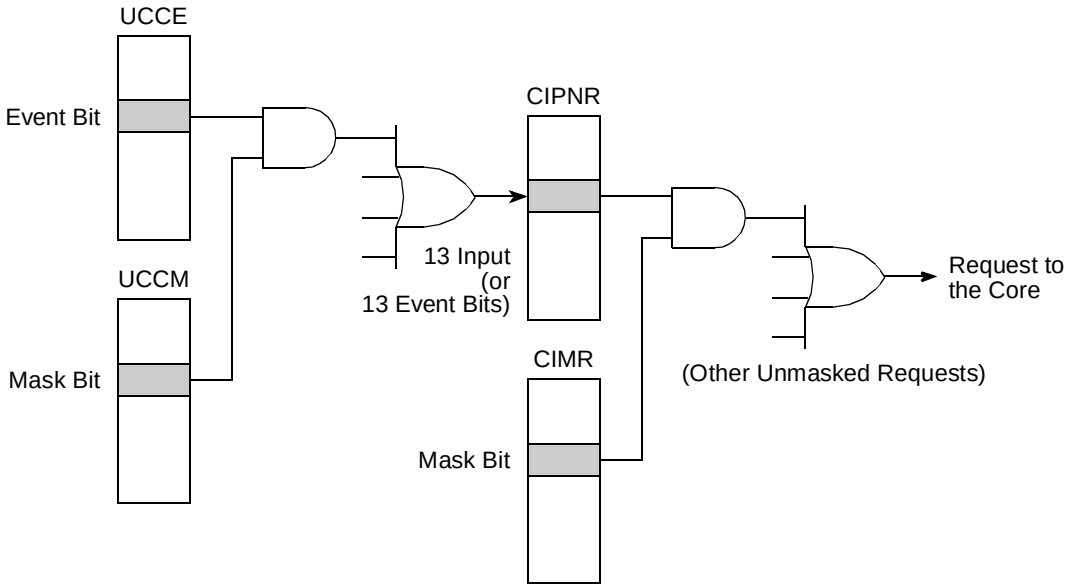


Figure 19-15. Interrupt Request Masking

19.2.6 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the CPU according to priority. The interrupt vector that allows the CPU to locate the interrupt service routine is made available to the CPU 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 19-12 lists encodings for the six low-order bits of the interrupt vector.

Table 19-12. 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
4	Reserved	0b00_0100
5	Reserved	0b00_0101
6	Reserved	0b00_0110
7	Reserved	0b00_0111

Table 19-12. Encoding the Interrupt Vector (continued)

Interrupt Number[1]	Interrupt Source Description	Interrupt Vector
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	Reserved	0b01_0000
17	PTP1	0b01_0001
19	Reserved	0b01_0011
20	VT	0b01_0100
21	RTC	0b01_0101
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	MCC1	0b10_0100
37	Reserved	0b10_0101
38	Reserved	0b10_0110
39	Reserved	0b10_0111
40	UCC5	0b10_1000
41	UCC6	0b10_1001
42	UCC7	0b10_1010
43	UCC8	0b10_1011
44	Reserved	0b10_1100
45	Reserved	0b10_1101
46	Reserved	0b10_1110
47	Reserved	0b10_1111

Table 19-12. Encoding the Interrupt Vector (continued)

Interrupt Number[1]	Interrupt Source Description	Interrupt Vector
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 19-12. The error vector is issued when no interrupt is requesting service.

19.3 Programming Model

The interrupt controller registers are described in the following sections:

- Section 19.3.1, “QUICC Engine System Interrupt Configuration Register (CICR)”
- Section 19.3.2, “QUICC Engine System Interrupt Control Register (CICNR)”
- Section 19.3.3, “QUICC Engine System RISC Interrupts Control Register (CRICR)”
- Section 19.3.4, “QUICC Engine System Interrupt Priority Register for WCC Peripherals (CIPWCC)”
- Section 19.3.5, “QUICC Engine System Interrupt Priority Register for XCC Peripherals (CIPXCC)”
- Section 19.3.6, “QUICC Engine System Interrupt Priority Register for YCC Peripherals (CIPYCC)”
- Section 19.3.7, “QUICC Engine System Interrupt Priority Register for ZCC Peripherals (CIPZCC)”
- Section 19.3.8, “QUICC Engine System Interrupt Priority Register for RISC Tasks A (CIPRTA)”
- Section 19.3.9, “QUICC Engine System Interrupt Priority Register for RISC Tasks B (CIPRTB)”
- Section 19.3.10, “QUICC Engine System Interrupt Pending Register (CIPNR)”
- Section 19.3.11, “QUICC Engine System Interrupt Mask Register (CIMR)”
- Section 19.3.12, “QUICC Engine RISC Interrupt Pending Register (CRIPNR)”
- Section 19.3.13, “QUICC Engine RISC Interrupt Mask Register (CRIMR)”
- Section 19.3.14, “QUICC Engine System Interrupt Vector Register (CIVEC)”
- Section 19.3.15, “QUICC Engine High System Interrupt Vector Register (CHIVEC)”

19.3.1 QUICC Engine System Interrupt Configuration Register (CICR)

CICR, shown in Figure 19-16, defines the highest priority interrupt and whether interrupts are grouped or spread according to the priority table, Table 19-10.

Offset 0x80 Access: Read/Write

	0	1	2–7	8	9	10	11	12	13	14	15	16–21	22	23	24–31
R	—		HP	—	GRTB	GRTA	—	GZCC	GWCC	GXCC	GYCC	—	HPIT		—
W															
Reset	All zeros														

Figure 19-16. QUICC Engine System Interrupt Configuration Register (CICR)

The CICR register bits are described in Table 19-13.

Table 19-13. 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.
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.

19.3.2 QUICC Engine System Interrupt Control Register (CICNR)

CICNR, shown in Figure 19-17, 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 19.3.1, "QUICC Engine System Interrupt Configuration Register (CICR)").

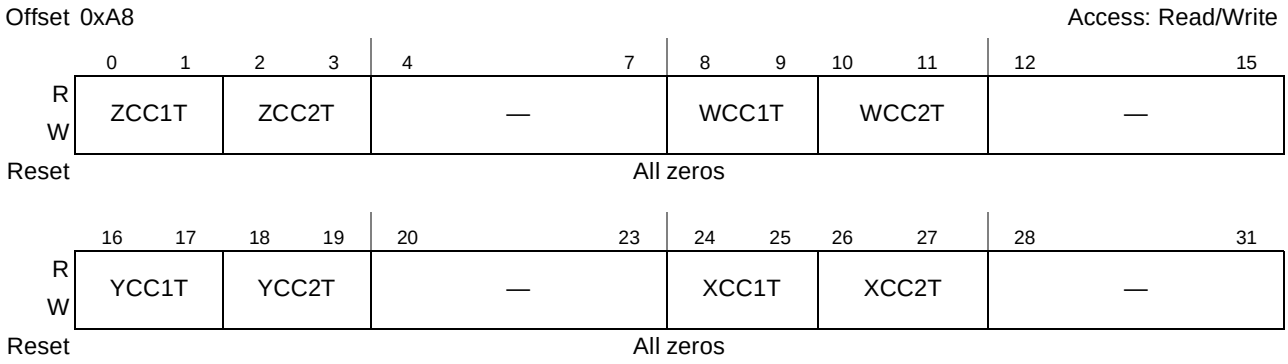


Figure 19-17. QUICC Engine System Internal Interrupt Control Register (CICNR)

Table 19-14 defines the bit fields of CICNR.

Table 19-14. 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 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 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
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 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 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

Table 19-14. CICNR Bit Settings (continued)

Bits	Name	Description
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 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 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 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 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

19.3.3 QUICC Engine System RISC Interrupts Control Register (CRICR)

CRICR, shown in Figure 19-18, defines the interrupt output type (QUICC Engine High or QUICC Engine Low) for RISC tasks interrupts.

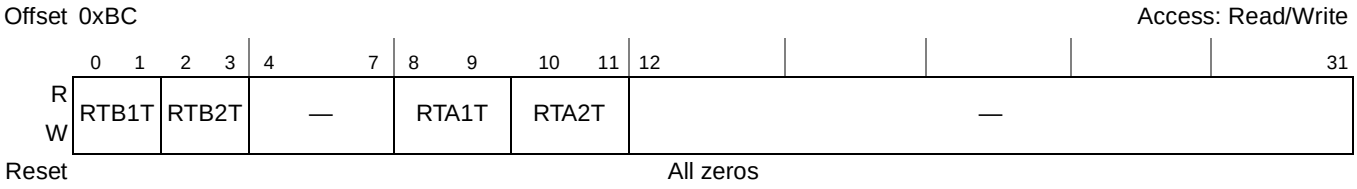


Figure 19-18. QUICC Engine System RISC Interrupts Control Register (CRICR)

Table 19-15 defines the bit fields of CRICR.

Table 19-15. 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 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 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 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 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.

19.3.4 QUICC Engine System Interrupt Priority Register for WCC Peripherals (CIPWCC)

CIPWCC, shown in Figure 19-19, defines the priority between Timers, SPIs, and external RISC interrupts.

Offset 0x98 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	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 19-19. QUICC Engine System Interrupt Priority Register for WCC (CIPWCC)

The CIPWCC register bits are described in Table 19-16.

Table 19-16. 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, MCC, 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.

19.3.5 QUICC Engine System Interrupt Priority Register for XCC Peripherals (CIPXCC)

The QUICC Engine system interrupt priority for XCC peripherals register (CIPXCC), shown in Figure 19-20, defines the priorities between the UCCs and MCC.

Offset 0x90 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	XCC1	XCC2	XCC3	XCC4	—	XCC5	XCC6	XCC7	XCC8	—
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 19-20. QUICC Engine System Interrupt Priority Register for XCC (CIPXCC)

Table 19-17 describes CIPXCC fields.

Table 19-17. CIPXCC Field Descriptions

Bits	Name	Description
0–2	XCC1	Priority order. These bits define which UCC, or the MCC, asserts its request in the XCC1 priority position. Note that each UCC, or the MCC, 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 MCC1 asserts its request in the XCC1 position. 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.

19.3.6 QUICC Engine System Interrupt Priority Register for YCC Peripherals (CIPYCC)

CIPYCC, shown in Figure 19-21, defines the priority of the UCCs.

Offset 0x94 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	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 1 0	1 1 1	0 0 0 0

Figure 19-21. QUICC Engine System Interrupt Priority Register for YCC (CIPYCC)

Table 19-18 describes the CIPYCC fields.

Table 19-18. 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 UCC6 asserts its request in the YCC1 position. 010 UCC7 asserts its request in the YCC1 position. 011 UCC8 asserts its request in the YCC1 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.

Table 19-18. CIPYCC Field Descriptions (continued)

Bits	Name	Description
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.

19.3.7 QUICC Engine System Interrupt Priority Register for ZCC Peripherals (CIPZCC)

CIPZCC, shown in Figure 19-22, defines the priority among a few peripherals.

Offset 0x9C Access: Read/ Write

	0–2	3–5	6–8	9–11	12–15	16–18	19–21	22–24	25–27	28–31
R	ZCC1	ZCC2	ZCC3	ZCC4	—	ZCC5	ZCC6	ZCC7	ZCC8	—
W										
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 19-22. QUICC Engine System Interrupt Priority Register for ZCC (CIPZCC)

The CIPZCC register bits are described in Table 19-19.

Table 19-19. 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.

19.3.8 QUICC Engine System Interrupt Priority Register for RISC Tasks A (CIPRTA)

CIPRTA, shown in Figure 19-23 defines the relative priority between the interrupt sources VT and SINT.

Offset 0xB0 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	RTA1	RTA2	RTA3	RTA4	—	RTA5	RTA6	RTA7	RTA8	—
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 19-23. QUICC Engine System Interrupt Priority Register for RISC Tasks A (CIPRTA)

The CIPRTA register bits are described in Table 19-20.

Table 19-20. 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 VT 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.

19.3.9 QUICC Engine System Interrupt Priority Register for RISC Tasks B (CIPRTB)

CIPRTB, shown in Figure 19-24 defines the relative priority among the interrupt sources EXT1, EXT2, EXT3 and EXT4.

Offset 0xB4 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	RTB1	RTB2	RTB3	RTB4	—	RTB5	RTB6	RTB7	RTB8	—
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 19-24. QUICC Engine System Interrupt Priority Register for RISC Tasks B (CIPRTB)

The CIPRTB register bits are described in Table 19-21.

Table 19-21. 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.

19.3.10 QUICC Engine System Interrupt Pending Register (CIPNR)

Each bit in the CIPNR, shown in Figure 19-25, corresponds to an interrupt source. When an interrupt is received, the interrupt controller sets the corresponding CIPNR bit.

Offset 0x8C Access: Read Only

	0	1	2	3	4	5	6	7
R	UCC1	UCC2	UCC3	UCC4	MCC1	—		
W								
Reset	All zeros							

	8	9	10	11	12	13	14	15
R	UCC5	UCC6	UCC7	UCC8	—	—	—	—
W								
Reset	All zeros							

	16	17	18	19	20	21		23
R	SPI2	SPI1	RTT	—				
W								
Reset	All zeros							

	24	25	26	27	28	29	30	31
R	—	SDMA	USB	Timer1	Timer2	Timer3	Timer4	—
W								
Reset	All zeros							

Figure 19-25. 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.

19.3.11 QUICC Engine System Interrupt Mask Register (CIMR)

Each bit in the CIMR, shown in Figure 19-26, 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 CPU.

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 CPU, according to its assigned priority. The CIMR can be read by the user at any time.

Offset 0xA0 Access: Read/Write

	0	1	2	3	4	5	7
R W	UCC1	UCC2	UCC3	UCC4	MCC1	—	
Reset	All zeros						

	8	9	10	11	12	15
R W	UCC5	UCC6	UCC7	UCC8	—	
Reset	All zeros					

	16	17	18	19	20	21	23
R W	SPI2	SPI1	RTT	—	—	—	
Reset	All zeros						

	24	25	26	27	28	29	30	31
R W	—	SDMA	USB	Timer1	Timer2	Timer3	Timer4	—
Reset	All zeros							

Figure 19-26. 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 CPU, 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.

19.3.12 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 20, "QUICC Engine Block Control." Figure 19-27 shows the CRIPNR fields.

Offset 0x88 Access: Read Only

	0	1	2	3	4	7
R	PTP1	—	PTP2	VT	RTC	—
W						
Reset	All zeros					

	8	9	10	11	12	15
R	EXT1	EXT2	EXT3	EXT4	—	
W						
Reset	All zeros					

	16	23
R	—	
W		
Reset	All zeros	

	24	31
R	—	
W		
Reset	All zeros	

Figure 19-27. CRIPNR Fields

Following is a short description of the RISC interrupt sources:

- **IEEE 1588 Interrupt (PTPx, RTC)** are interrupts to the CPU that are asserted by the QUICC Engine module while it is processing an interrupt request from PTPn_TMR_PEVENT or TMR_TEVENT. This mechanism allows for the QUICC Engine RISC to interrupt the host CPU due to certain events which occur while the QUICC Engine UCC's Time Stamp Unit processes a PTP packet or upon detection of an IEEE 1588 Timer event. See Section 31.8, "Time Stamp Unit Mode Registers," and Section 31.9, "IEEE1588 Timer Mode Registers."

- **External Request Interrupt (EXT*x*)** is an interrupt to the CPU 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 CPU 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 20.5, “QUICC Engine External Requests” in Chapter 20, “QUICC Engine Block Control.”
- **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, e.g. 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 20.3.3, “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 20.6, “Multi-Threading” for more details.

19.3.13 QUICC Engine RISC Interrupt Mask Register (CRIMR)

CRIMR, shown in Figure 19-28, is used to mask interrupts that are pending in CRIPNR. Setting a bit in CRIMR masks the corresponding pending interrupt in CRIPNR.

Offset 0xA4 Access: Read/Write

	0	1	2	3	4	5	7
R W	PTP1	—	PTP2	VT	RTC	—	
Reset	All zeros						

	8	9	10	11	12	15
R W	EXT1	EXT2	EXT3	EXT4	—	
Reset	All zeros					

	16	23
R W	—	
Reset	All zeros	

	24	31
R W	—	
Reset	All zeros	

Figure 19-28. CRIMR Fields

19.3.14 QUICC Engine System Interrupt Vector Register (CIVEC)

CIVEC, shown in Figure 19-29, 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 19.2.6, “Interrupt Vector Generation and Calculation,” for the list of codes.

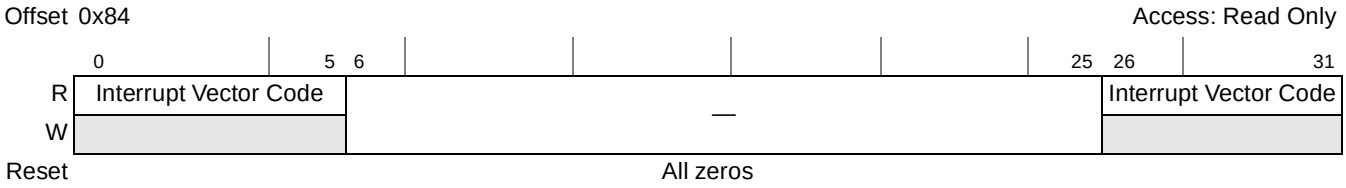


Figure 19-29. QUICC Engine System Interrupt Vector Register (CIVEC)

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 19-30.

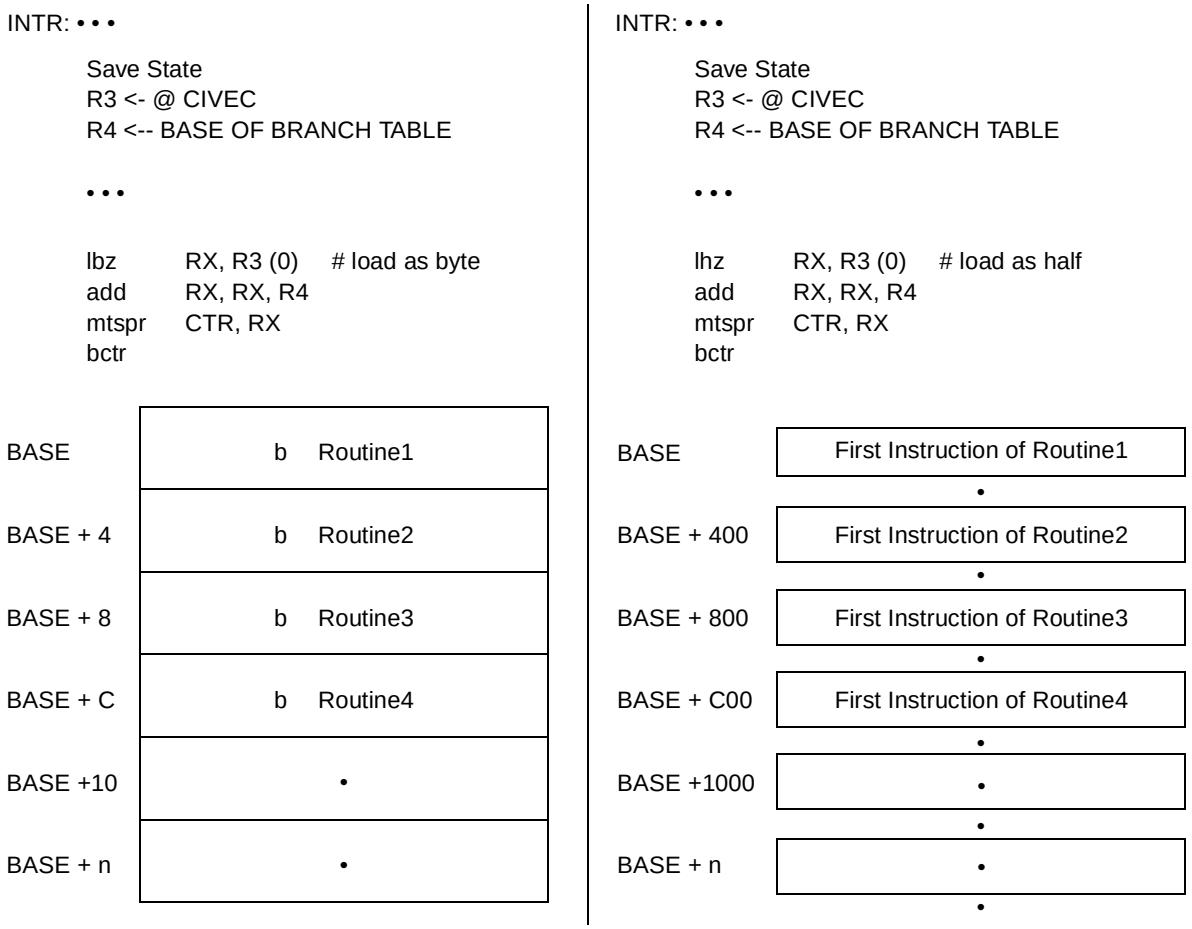


Figure 19-30. Interrupt Table Handling Example

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.

19.3.15 QUICC Engine High System Interrupt Vector Register (CHIVEC)

CHIVEC, shown in Figure 19-31, 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 19.2.6, “Interrupt Vector Generation and Calculation,” for the list of codes.

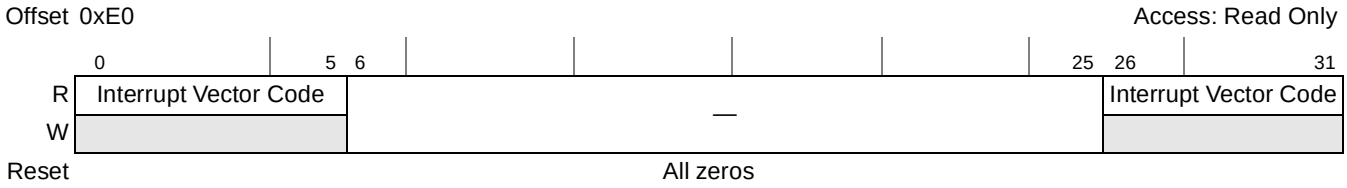


Figure 19-31. QUICC Engine High System Interrupt Vector Register (CHIVEC)

Chapter 20
QUICC Engine Block Control

20.1 Introduction

The QUICC Engine control registers 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. The commands are issued by writing to the QUICC Engine command register (CECR). These commands are used to initialize the RISC(s) and to control the process of each peripheral controller (UCC, MCC, or SPI) while the RISC engines are running.

See Chapter 2, "Memory Map."

20.2 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, MCC, 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. The user can override the default values with other values by issuing the ASSIGN PAGE Command to the QUICC Engine block. See Section 20.3.1.1.1, "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 20-1 depicts the default values of the parameter RAM Pages base addresses assigned by the QUICC Engine block to the different peripherals.

Table 20-1. 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 20-1. Default Parameter RAM Base Addresses (continued)

Address	Peripheral	Size (Bytes)
0x8000	UCC5 (Rx and Tx)	256
0x8100	UCC6 (Rx and Tx)	256
0x8200	UCC7 (Rx and Tx)	256
0x8700	MCC (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
0x8300	UCC8 (Rx and Tx)	256

20.3 QUICC Engine Control Registers

20.3.1 QUICC Engine Command Register (CECR)

The CECR register is used by the CPU 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 CPU sets the CECR[FLG] bit (see Figure 20-1) when it issues a command, and the QUICC Engine block clears the FLG after execution, indicating to the CPU 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 20-2 below.

Table 20-2. Mapping of CPM FCCs and SCCs to QUICC Engine Block

CPM Peripheral	QUICC Engine Peripheral
FCC1	UCC1
FCC2	UCC2
FCC3	UCC3
SCC1	UCC5
SCC2	UCC6
SCC3	UCC7
SCC4	UCC8

Offset QUICC Engine base address + 0x00100 Access: Read/Write

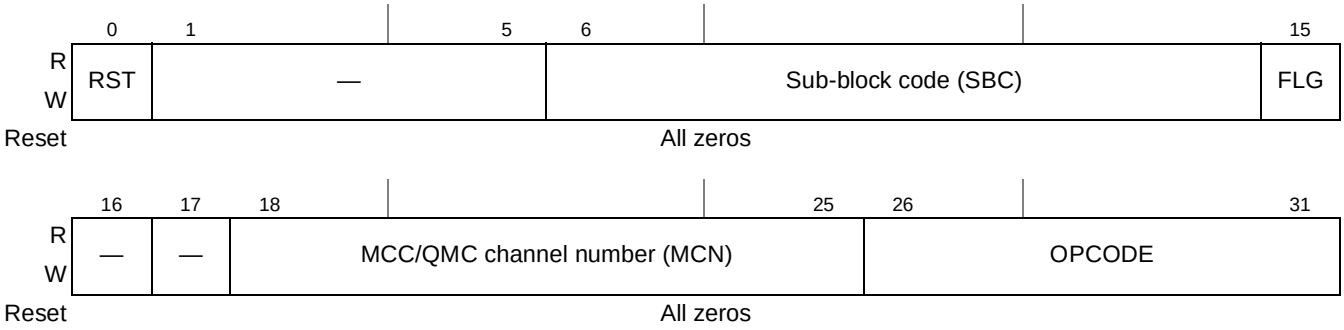


Figure 20-1. QUICC Engine Command Register (CECR)

Table 20-3 describes CECR fields.

Table 20-3. QUICC Engine Command Register Field Descriptions

Bit	Name	Description
0	RST	Software reset command. Set by the CPU 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 is issued. RST is useful for resetting the registers and parameters for all the channels (UCCs, SPIs, MCC) 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 20-3. 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 CPU 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>UCC2
Fast 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>UCC6
Fast Protocols Mode</td><td>001010000</td><td>UCC6
Slow Protocols Mode</td></tr>
<tr><td>101100000</td><td>UCC7
Fast Protocols Mode</td><td>001100000</td><td>UCC7
Slow Protocols Mode</td></tr>
<tr><td>101110000</td><td>UCC8
Fast Protocols Mode</td><td>001110000</td><td>UCC8
Slow Protocols Mode</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>MCC</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 CPU 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>
</table>

Table 20-3. QUICC Engine Command Register Field Descriptions (continued)

Bit	Name	Description
18–25	MCN	MCC/QMC Channel Number. Specifies the channel number in the case of an MCC/QMC command. In UCC protocols, this field contains the protocol code as follows: 0x00 HDLC/transparent 0x0A ATM 0x0C Ethernet 0x0D L2 Switch 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.
26–31	OPCODE	Operation code. Settings are listed in Table 20-4.

20.3.1.1 QUICC Engine Commands

Table 20-4 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 00000000, the UCCx, regardless of what mode it is in, performs an Rx and Tx initialization of all parameters, as does the SPI and the MCC. However, as the table shows, this opcode has no affect on the Timer.

As another example, if the opcode is set to 000111, and if the UCC is in Slow Protocols Mode, the Receive Buffer Descriptors (RxBD) are closed for both the UCC and SPI, and the MCC is reset. 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 23, “Unified Communications Controllers (UCCs),” for more information on the UCC: Fast Protocols and Slow Protocols modes.

Table 20-4. QUICC Engine Command Opcodes

Opcode *(command)*	Affect of Opcode on Indicated Element				
	UCC*n* Protocols: Ethernet, HDLC/bus, Transparent, ATM **(Fast Protocols Mode)**	**UCC*n* Protocols: UART, BISYNC, QMC/SAM** **(Slow Protocols Mode)**	**SPI*n***	**MCC**	**Timer**
000000	INIT RX AND TX PARAMS	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	INIT RX PARAMS	—
000010	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	—
000011	ENTER HUNT MODE	ENTER HUNT MODE	—	INIT MCC RX AND TX PARAMS (16 BIT)	—
000100	STOP TX	STOP TX	—	MCC STOP TRANSMIT	—
000101	GRACEFUL STOP TX	GRACEFUL STOP TX	—	INIT MCC TX PARAMS (ONE CHANNEL)	—
000110	RESTART TX	RESTART TX	—	INIT MCC RX PARAMS (ONE CHANNEL)	—
000111	L2SWITCH COMMAND	—	—	MCC RESET	—

Table 20-4. QUICC Engine Command Opcodes (continued)

Opcode *(command)*	Affect of Opcode on Indicated Element				
	UCC*n* Protocols: Ethernet, HDLC/bus, Transparent, ATM (Fast Protocols Mode)	**UCC*n* Protocols: UART, BISYNC, QMC/SAM (Slow Protocols Mode)**	**SPI*n***	**MCC**	**Timer**
001000	SET GROUP ADDRESS	—	—	—	SET TIMER
001001	INSERT CELL	—	—	MCC STOP RECEIVE	—
001010	ATM TRANSMIT COMMAND	RESET BCS	—	USB STOP TX	—
001011	CELL POOL GET	—	—	PPP: INIT PPP TX LINK	—
001100	CELL POOL PUT	QMC STOP TX	—	USB RESTART TX	—
001101	IMA HOST COMMAND	QMC STOP RX	—	PPP: INIT TX BACKEND	—
001110	—	—	—	PPP: INIT MUX PROCESS	—
001111	PUSHSCHED COMMAND	PUSHSCHED COMMAND	PUSHSCHED COMMAND	PUSHSCHED COMMAND	PUSHSCHED COMMAND
010000	Reserved				
010001	ATM MULTITHREAD INIT	—	—	RESET SU FILTER	—
010010	ASSIGN PAGE	ASSIGN PAGE	ASSIGN PAGE	ASSIGN PAGE	ASSIGN PAGE
010011	SET LAST RECEIVE REQUEST THRESHOLD	—	—	SET ENTRY IN HASH LOOKUP TABLE (ETHERNET)	—
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	ASSIGN PAGE TO DEVICE
011010	GRACEFUL STOP RECEIVE	—	—	—	—
011011	RESTART RECEIVE	—	—	INIT PPP RX LINK	—
011100	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.				
011101	—	—	—	PPP: INIT RX BACKEND	—
011110	—	—	—	PPP: INIT DEMUX PROCESS	—
011111–101010	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.				
101011	—	—	—	INIT PPP RX AND TX LINK	—
101100	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.				
101101	—	—	—	PPP: INIT RX AND TX BACKEND	—
101110	—	—	—	INIT PPP MUX PROCESSES: MUX AND DEMUX	—
001101–111111	Undefined. Reserved for Use by Freescale-supplied RAM Microcodes.				

If a reserved command is issued, the QUICC Engine block enters 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 20-4 are described in Table 20-5.

Table 20-5. 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 MCC RX AND TX PARAMS — ONE CHANNEL	Initialize receive and transmit parameters. Initializes the receive and transmit parameters of the peripheral controller. Differs from INIT RX AND TX PARAMS in that, for the MCCs, issuing INIT RX AND TX PARAMS initializes 32 consecutive channels beginning with the channel number specified in CECR[MCN], but issuing INIT MCC RX AND TX—ONE CHANNEL initializes only the channel in the command; For more information see Section 34.3, “MCC Commands.”
INIT RX PARAMS	Initialize receive parameters. Initializes the receive parameters of the peripheral controller. Note that for the MCC, issuing this command initializes 32 channels at a time. For more information see Section 34.3, “MCC Commands.”
INIT MCC RX PARAMS— ONE CHANNEL	Initialize MCC receive parameters for only a single channel according to MCC channel number field. For more information see Section 34.3, “MCC Commands.””
INIT TX PARAMS	Initialize transmit parameters. Initializes the transmit parameters of the peripheral controller. Note that for the MCC, issuing this command initializes 32 channels at a time; For more information see Section 34.3, “MCC Commands.”
INIT TX PARAMS— ONE CHANNEL	Initialize MCC transmit parameters for only a single channel according to MCC channel number field. For more information see Section 34.3, “MCC Commands.”
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.
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 20.4.5, “set timer Command”.

Table 20-5. Command Descriptions (continued)

Command	Description
SET GROUP ADDRESS	Set group address. Sets a bit in the hash table for the Ethernet logical group address recognition function. See Table 30-95.
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 30.8, "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.
MCC STOP TRANSMIT	See Section 34.3, "MCC Commands."
MCC STOP RECEIVE	See Section 34.3, "MCC Commands.""
MCC RESET	MCC reset. Provides a hard reset to the MCC FIFOs. For more information see Section 34.3, "MCC Commands." To use this command, software should execute the following sequence: 1 Disable the TDM by clearing the appropriate enable bit in SIGMR[4–7, 12–15] (See Table 36-7). 2 Issue the MCC RESET command. 3 Issue the INIT RX AND TX command. 4 Reprogram the specific MCC channel, global parameters, and any BDs that need to be updated. 5 Set the appropriate enable bit in SIGMR[4–7, 12–15]. (See Table 36-7).
RESET SU FILTER	This command resets the filtering algorithm to ensure that the next SU will be received, even if it would normally have been filtered. This command could be issued periodically so that the CPU can check to make sure that the link is really up and not simply receiving flags.
ATM TRANSMIT	See Section 32.4.1, "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.
INIT PPP RX LINK	Initializes the correct state for ML MC receiver operation. MCN should be pointed to the channel which is mapped to the LPT. The host should issue this command for each link in the ML MC system. It should be issued AFTER the INIT Rx command.
INIT PPP TX LINK	Initializes the correct state for ML MC transmitter operation. MCN should be pointed to the channel which is mapped to the LPT. The host should issue this command for each link in the ML MC system. It should be issued AFTER the INIT Tx command.
INIT PPP RX & TX LINK	This command combines both previous commands.
INIT PPP RX BACK-END	Initializes the Rx back-end tasks of the PPP. This command initializes the Rx packet reconstruction task.
INIT PPP TX BACK-END	Initializes the Tx back-end tasks of the PPP. This command initializes the Tx fragmentation process.
INIT PPP BACK-END	Initializes the Tx back-end and the Rx back-end tasks. Combines both previous commands
INIT DE-MUX PROCESS	Initializes the Rx demultiplexing process This command should be issued following the Back-end initialization
PPP INIT MUX PROCESS	Initializes the Tx MUX encapsulation process.

Table 20-5. Command Descriptions (continued)

Command	Description
INIT PPP MUX PROCESSES	Initialize both, the Tx MUX encapsulation process and the receive de-mux process.
INSERT CELL	See Section 39.20.2, "MSP Insert Cell Command."
CELL POOL GET	See Section 39.20.1, "MSP FCP Commands."
CELL POOL PUT	See Section 39.20.1, "MSP FCP Commands.
PUSHSCHED	Modify the start address for the task running on a given peripheral
ATM MULTI THREAD INIT	See Section 32.4.1.3, "ATM Multi-Thread Init."
ASSIGNING PAGE	See Section 20.3.1.1.1, "Assign Page Command".
GRACEFUL STOP RECEIVE	See Section 30.8, "Ethernet Command Set".
RESTART RECEIVE	See Section 30.8, "Ethernet Command Set".

20.3.1.1.1 Assign Page Command

This command is used to modify the default value of the Parameter RAM Base Address for a given Peripheral (UCC, MCC, 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 20.7, "Serial Number (SNUM)," for a description of the term SNUM and the values to be used.

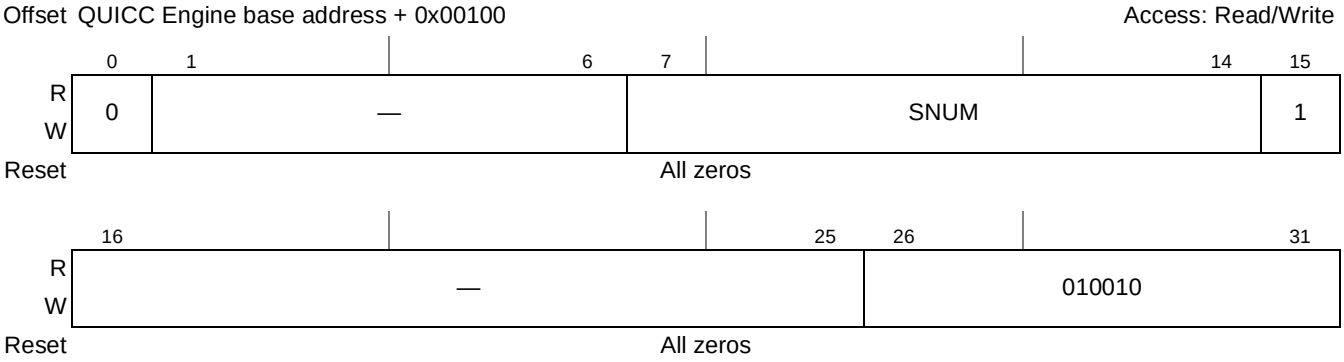


Figure 20-2. CECR for Assign Page Command

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 than assign the 'Lowest' SNUM (See Table 20-15) 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 20-15). Alternatively it is possible to issue the Assign Page to Device command only once (See Section 20.3.1.1.2, "Assign Page to Device").

20.3.1.1.2 Assign Page to Device

This command is used to modify the default value of the Parameter RAM base address for a given Peripheral (UCC, MCC, 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.

20.3.1.1.3 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 request is enabled by setting CECR[17].

Offset QUICC Engine base address + 0x00100 Access: Read/Write

	0	1–6	7–14	15
R	0	—	SNUM	1
W				
Reset	All zeros			

	16	17	18–25	26–31
R	—	—	—	001111
W				
Reset	All zeros			

Figure 20-3. CECR for PushSched Command

20.3.2 QUICC Engine Command Data Register (CECDR)

This register is used to transfer data as a result of the command written to the CECR register. If the CECDR data is used by the issued command, the CPU must write the data to the CECDR register before setting the CECR[FLG] bit. The CECDR may be updated again only after CECR[FLG] is cleared by the RISC.

Offset QUICC Engine base address + 0x00108 Access: Read/Write

	0							31
R W	Command Related Data							
Reset	All zeros							

Figure 20-4. QUICC Engine Command Data Register (CECDR)

Refer to specific protocol commands for the contents of this register (e.g. Section 30.8.1, “Init Tx, Init Rx and InitTx and Rx Parameters Command”).

20.3.3 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 20.6, “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 Access: Read/Write
CEVTMR: QUICC Engine base address + 0x00134

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VT0	VT1	VT2	VT3	VT4	VT5	VT6	VT7	VT8	VT9	VT10	VT11	VT12	VT13	—	
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	VT16	VT17	VT18	VT19	VT20	VT21	VT22	VT23	VT24	VT25	VT26	VT27	VT28	VT29	—	
Reset	All zeros															

Figure 20-5. Virtual Task Event/Mask Register (CEVTER/CEVTMR)

Table 20-6. CEVTER/CEVTMR Field Descriptions

Bits	Name	Description
0–13	VT*n*	Virtual Tasks 0–13
14–15	—	Reserved
16–29	VT*n*	Virtual Task 16–29
30–31	—	Reserved

20.3.4 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, to put them in a known, cleared state. After the initialization, ECC mode can be enabled. Figure 20-6. shows the register containing the ECC control bits.

Offset QUICC Engine base address + 0x00138 Access: Read/Write

	0	1	2	3	4	5	15
R	MEE	IEE	—		CIR	—	
W							
Reset	All zeros						

Figure 20-6. QUICC Engine RAM Control Register (CERCR)

Table 20-7. 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 (may affect performance).
1	IEE	Instruction RAM ECC enable 0 ECC disabled in the instruction RAM. 1 ECC enabled in the instruction RAM (may affect performance).
2–3	—	Reserved, should be cleared.
4	CIR	Common Instruction RAM 0 Each of the two RISC processors has its own 24-Kbyte instruction RAM. Performance may be better since RISCs are not competing on a common resource when fetching instructions. When CIR=0, the instruction ram is seen through IADD as follows: RISC0 0x8_0000–0x8_5FFF RISC1 0x8_8000–0x8_DFFF In case the two RISCs should run the same code, the code should be duplicated in the regions for RISC0 and RISC1. 1 Both RISC processors are sharing a common 48-Kbyte instruction RAM (must be used when instruction RAM code is more than 24 Kbyte). When CIR=1, the instruction RAM is seen as a consecutive 48-Kbyte region at addresses 0x80000–0x8BFFF.
5–15	—	Reserved, should be cleared.

20.3.5 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.

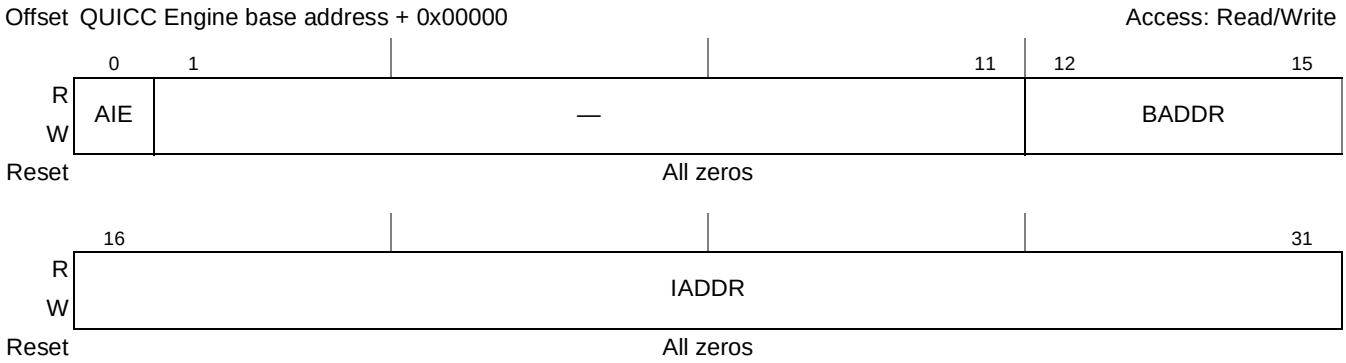


Figure 20-7. IRAM Address Register (IADD)

IADD field descriptions are shown in Table 20-8.

Table 20-8. 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 0x0 to 0xBFFC.

20.3.6 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.

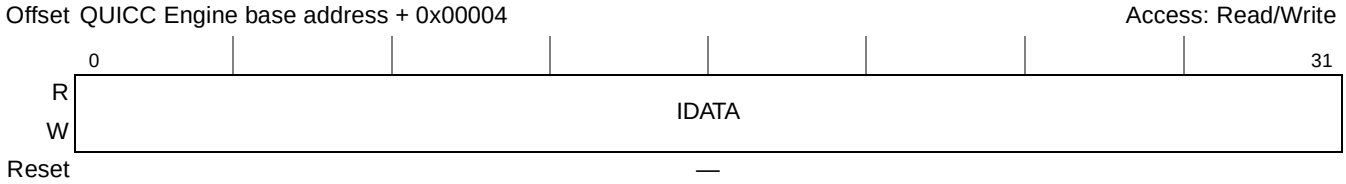


Figure 20-8. IRAM Data Register (IDATA)

Table 20-9. IDATA Field Descriptions

Bits	Name	Description
0–31	IDATA	Data written to or read from the I-RAM at the specified IADD[IADDR] address.

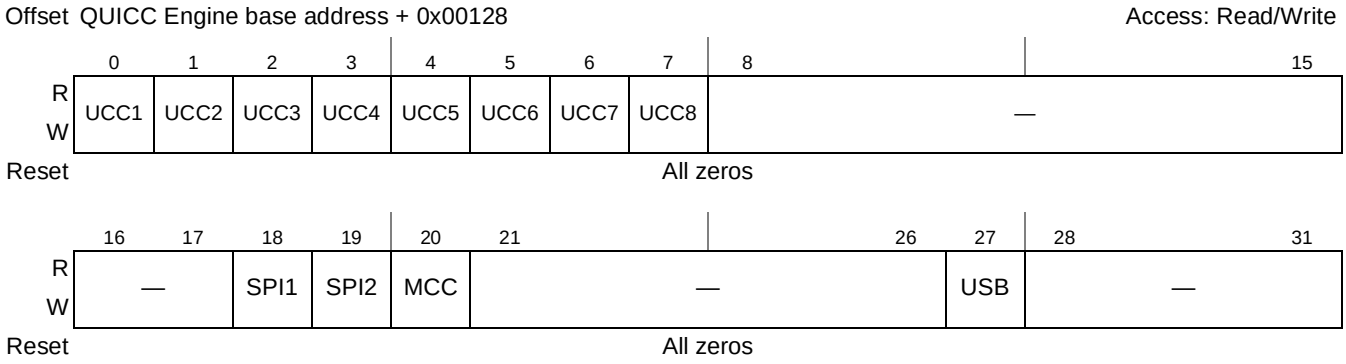


Figure 20-9. QUICC Engine Serial Interrupt Request Mask Register (CESIMR)

20.3.7 QUICC Engine Microcode Revision Number

The CEURNR register is a 32 bit read only register which contains the microcode revision number.

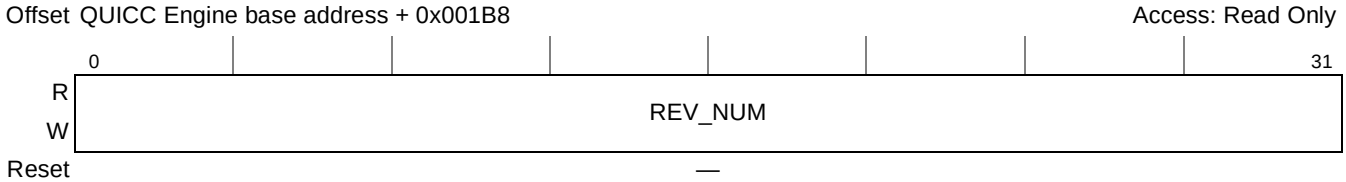


Figure 20-10. QUICC Engine Microcode Revision Number Register (CEURNR)

Table 20-10. CEURNR Field Descriptions

Bits	Name	Description
0–31	REV_NUM	Microcode revision number, value 0xCE00_0000.

20.3.8 QUICC Engine Controller Configuration Register (CECCR)

The QUICC Engine controller configuration register (CECCR), as shown in Figure 20-11, 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 engines. 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.

Offset QUICC Engine base address + 0x00104 Access: Read/Write

	0	1	2	7	8	9	10	15
R W	TIME	TEC	TIMEP		ERM1	ERM2	—	
Reset	All zeros							

	16	19	20	21	22	23	24	25	26	31
R W	—		EDM1	EDM2	EDM3	EDM4	ERM3	ERM4	—	
Reset	All zeros									

Figure 20-11. QUICC Engine Controller Configuration Register (CECCR)

CECCR bit fields are described in Table 20-11.

Table 20-11. 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 21.5.1, “CMX General Clock Route Register (CMXGCR),” on page 21-9. 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.

20.3.9 QUICC Engine Time-Stamp Control Register (CETSCR)

The QUICC Engine time-stamp control register (CETSCR), shown in Figure 20-12, configures the QUICC Engine time-stamp timers (CETSR1,2). The time-stamp timers are used by the ATM and the HDLC controllers. CETSR2 is used by the Ethernet controller.

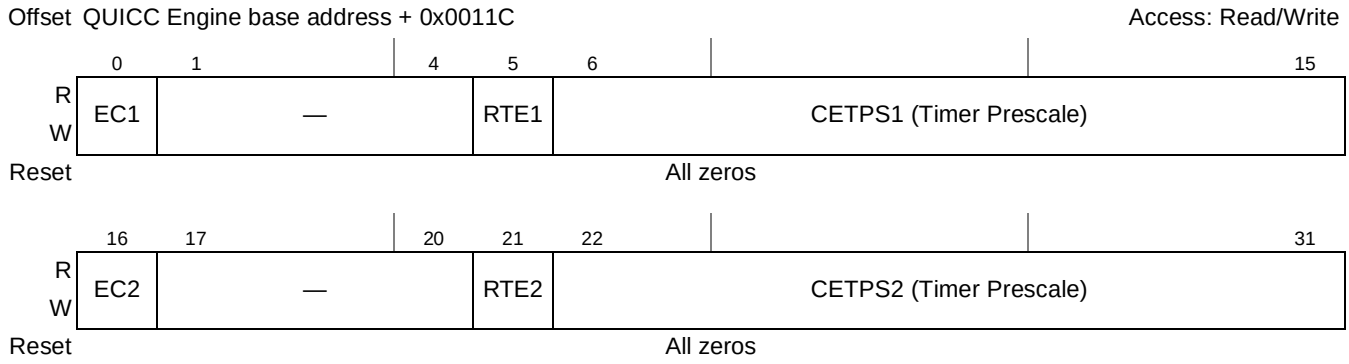


Figure 20-12. QUICC Engine Time-Stamp Control Register (CETSCR)

Table 20-12 describes CETSCR fields.

Table 20-12. CETSCR Field Descriptions

Bits	Name	Description
0, 16	EC1 EC2	External clock 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, 17–20	—	Reserved
5, 21	RTE1 RTE2	Time stamp enable. 0 Disable time-stamp timer. 1 Enable time-stamp timer.
6–15, 22–31	CETP1 CETP2	Time-stamp timer pre-scale. Must be programmed to generate a 1 micro second period input clock to the time-stamp timer. (Time-stamp frequency = (QUICC Engine frequency)/(CETPS+2))

20.3.10 QUICC Engine Time-Stamp Registers (CETSR*n*)

The QUICC Engine time-stamp registers (CETSR*n*), shown in Figure 20-13, contain the time stamp for time stamp 1 and time stamp 2 respectively.

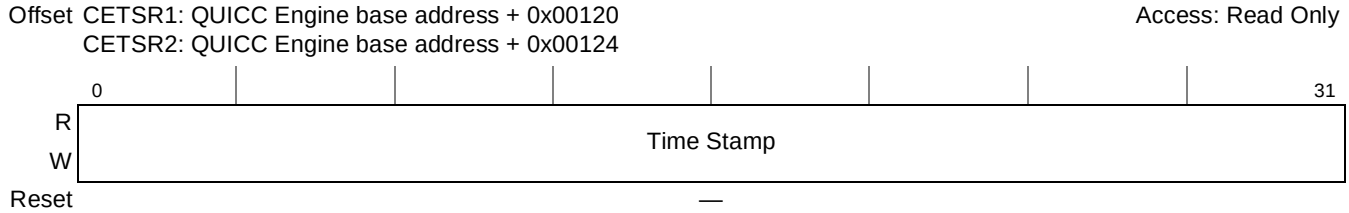


Figure 20-13. QUICC Engine Time-Stamp Registers (CETSR*n*)

After reset, setting CETSCR[RTE1, RTE2] causes the time stamp to start counting microseconds from zero.

20.4 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 CPU 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 (e.g. 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 20.3.8, "QUICC Engine Controller Configuration Register (CECCR)." The trigger value is a multiple of 1,024 internal timer's clocks (see Figure 20-11).

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 20.4.4, "RISC Timer Event Register (CETER)/Mask Register (CETMR)"), and by issuing SET TIMER to the CECR, see Section 20.3.1, "QUICC Engine Command Register (CECR)".

20.4.1 RISC Timer Table Parameter RAM

Two areas of multi-user RAM, shown in Figure 20-14, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

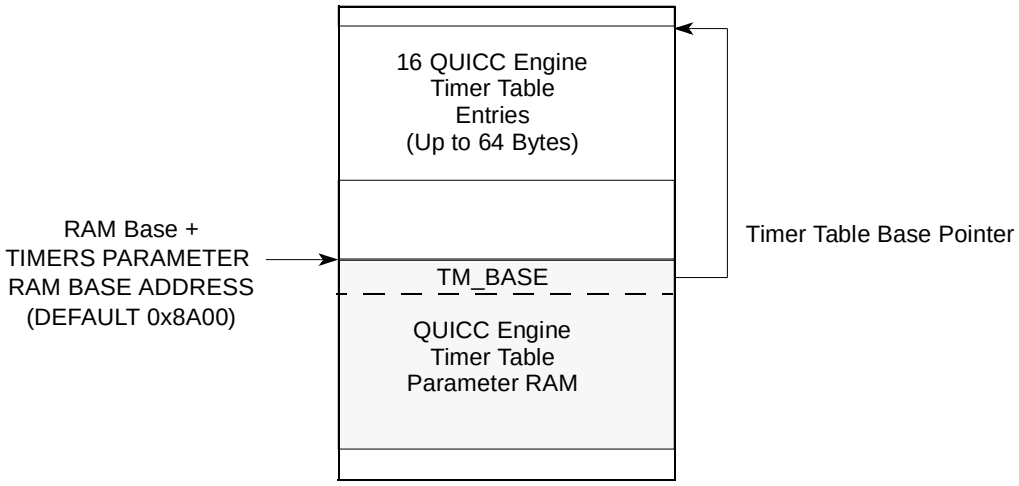


Figure 20-14. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 20-13.

Table 20-13. RISC Timer Table Parameter RAM

Offset[1]	Name	Description
0x00	TM_BASE	RISC timer table base address. The actual timers are a small block of memory in the 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	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	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	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 20-13. RISC Timer Table Parameter RAM (continued)

Offset[1]	Name	Description
0x08	TM_CMD	RISC timer command register. Used as a parameter location when the SET TIMER command is issued. The user should write this location before issuing the set timer command. This register is defined in Section 20.4.2, "RISC Timer Command Register (TM_CMD)."
0x0C	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)

20.4.2 RISC Timer Command Register (TM_CMD)

Figure 20-15 shows the RISC timer command register (TM_CMD).

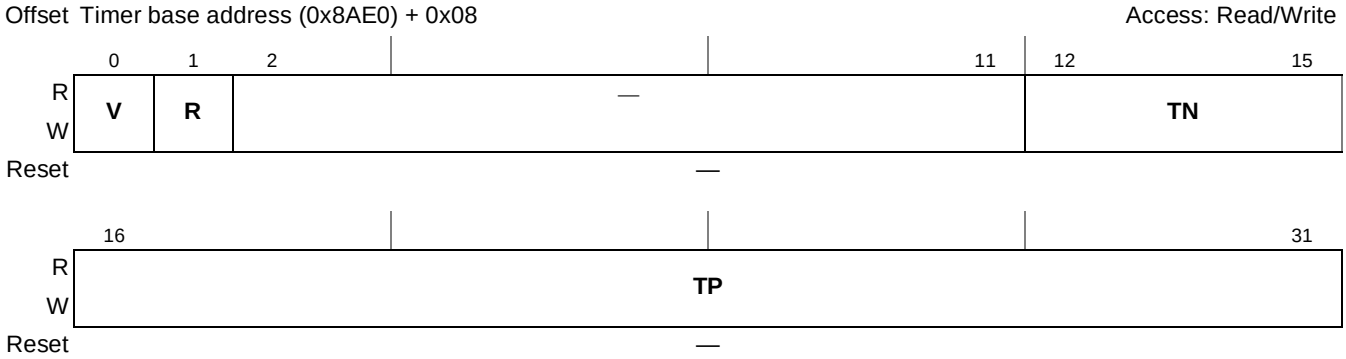


Figure 20-15. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 20-14.

Table 20-14. 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.

Table 20-14. TM_CMD Field Descriptions (continued)

Bits	Name	Description
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.

20.4.3 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.

20.4.4 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 19.3.11, “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 Access: Read/Write
CETMR: QUICC Engine base address + 0x0011A

	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 20-16. RISC Timer Event Register/Mask Register (CETER/CETMR)

20.4.5 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 20.4.2, “RISC Timer Command Register (TM_CMD).”

20.4.6 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.

20.4.7 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.

20.5 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 20.3.8, "QUICC Engine Controller Configuration Register (CECCR)," for more details. Events related to the external requests may cause assertion of an interrupt request to the CPU. These events are managed through the CEEXE*n* and CEEXM*n* registers, see Section 20.5.1 below. The external requests resources may be used by various protocols. The L2 switch uses these resources. See Chapter 40, "L2 Ethernet Switch," for details.

20.5.1 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 CPU. The mask register is programmed by the CPU 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 20-17. QUICC Engine External Event and Mask Registers (CEEXE*n*/CEEXM*n*)

20.6 Multi-Threading

The UCC Ethernet Controller and the UCC ATM Controller are able to processes frames or cells at high bit rates (gigabit Ethernet and OC-12 nominal 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 20-18 depicts the multithreading architecture at a high level.

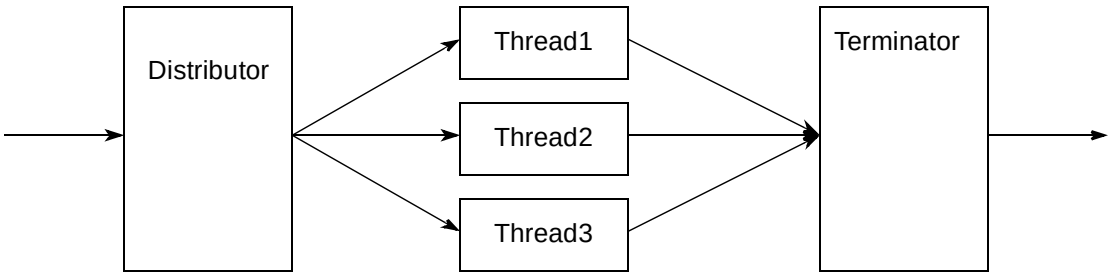


Figure 20-18. Multi-Threading Processing Mechanism

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 30.8.1, “Init Tx, Init Rx and InitTx and Rx Parameters Command,” on page 30-115 and Section 32.3.3, “Multi-Threading Structures,” on page 32-76.

Table 20-15 lists the Serial Numbers (SNUMs) to be used to program the multithreading options in the UCCs for Ethernet and ATM.

20.7 Serial Number (SNUM)

Each peripheral has a unique serial number (SNUM) associated with it. Threads, which are used by the multithreading mechanism, see Section 20.6, “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 20.3.1.1.1, “Assign Page Command”
- When initializing the multithreading mechanism

Table 20-15 shows the unique SNUM for each peripheral and thread. The names under ‘Serial Name’ are Peripheral or Thread names.

Table 20-15. SNUM Table

SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name
0x00	UCC1 TX	0x40	UCC5 TX	0x80	—	0xC0	MCC TX
0x01	UCC1 RX	0x41	UCC5 RX	0x81	—	0xC1	MCC RX
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	UCC6 TX	0x90	USB TX	0xD0	—
0x11	UCC2 RX	0x51	UCC6 Rx	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	UCC7 TX	0xA0	SPI1 TX	0xE0	—
0x21	UCC3 RX	0x61	UCC7 RX	0xA1	SPI1 RX	0xE1	—
0x24	**Thread24**	0x64	—	0xA4	—	0xE4	—
0x25	**Thread25**	0x65	—	0xA5	—	0xE5	—
0x28	—	0x68	—	0xA8	**Thread4**	0xE8	**Thread12**

Table 20-15. SNUM Table (continued)

SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name	SNUM	Serial Name
0x29	—	0x69	—	0xA9	**Thread5**	0xE9	**Thread13**
0x2C	**Thread26**	0x6C	—	0xAC	—	0xEC	—
0x2D	**Thread27**	0x6D	—	0xAD	—	0xED	—
0x30	UCC4 TX	0x70	UCC8 TX	0xB0	SPI2 Tx	0xF0	TIMER
0x31	UCC4 RX	0x71	UCC8 RX	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 20.3.1.1.1, “Assign Page Command” for usage of this SNUM.

Chapter 21
QUICC Engine 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, UPCs, 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 1-16) and a bank of external clocks (1-24) from the parallel I/O ports.
- 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 21.5.5, "CMX UCC Clock Route Register (CMXUCR1)," for specific details on setting this mode.
- 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 SMI master.

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 21-1 shows a block diagram of the CMX.

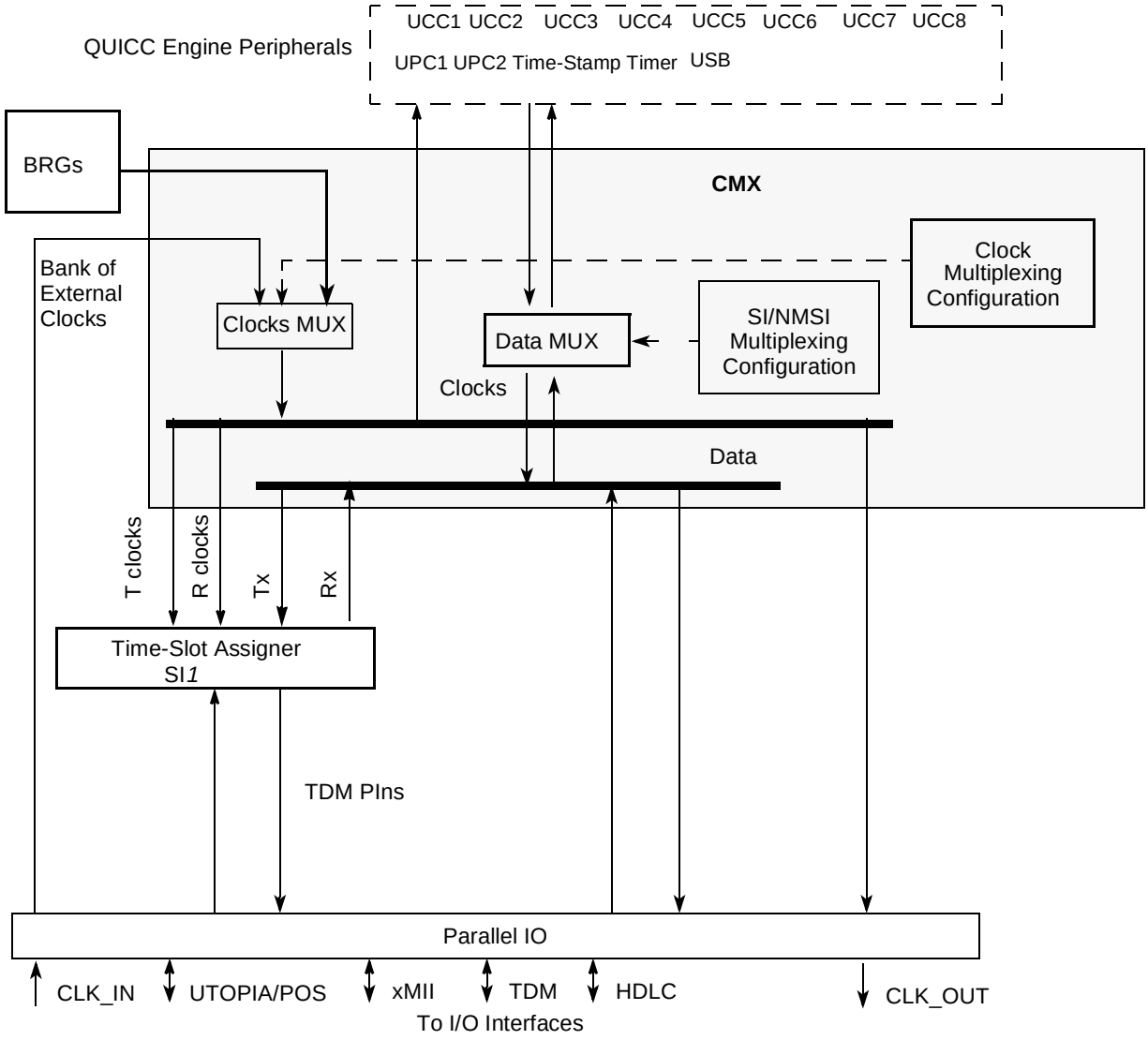


Figure 21-1. QUICC Engine Multiplexing and Timers Logic (CMX) Block Diagram

21.1 Working with Peripherals in NMSI Mode

In NMSI mode, the QUICC Engine MUX is primarily used for clock assignment. The clocks can be assigned from an external bank of twenty four clock pins or a bank of sixteen BRGs, to the following peripherals:

- UCC 1–8
- UPC 1–2 (Note that UCCs routed to the UPCs do not need to be assigned through the QUICC Engine MUX).
- Time stamps and Timer.
- USB

- UPC internal rate clock.
- Selection of the SMI (Serial Management Interface) master from the UCCs.

21.2 Working with TDM

When working with TDM, the QUICC Engine MUX requires clock and sync to be assigned to the SI. See specific details in the CMXSI1CRL, CMXSI1CRH and CMXSI1SYR registers. The clocks and syncs can be assigned from an external bank of twenty-four clock pins or a bank of sixteen BRGs to TDM A through H.

21.3 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 21-2. Once connections are made to the TSA, the clock is routed to UCCs directly from the TSA through the QUICC Engine Mux and clock assignment to those UCCs is not required. Exact time slot assignment routing decisions are made in the SI RAM.

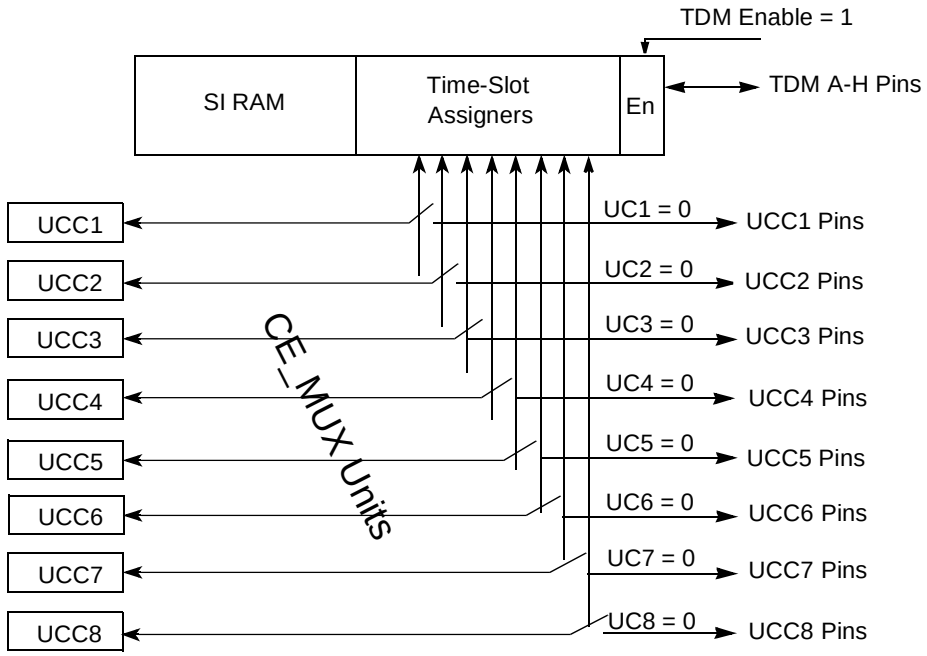


Figure 21-2. Enabling Connections to the TSA

21.4 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 CMXUCRx[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 21-3. The clock routing options are described in Table 21-1 and Table 21-2. The color coding is: Yellow indicates common clocks, green indicates clocks that are routable to the BRGs, white indicate normal multiple options. 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.

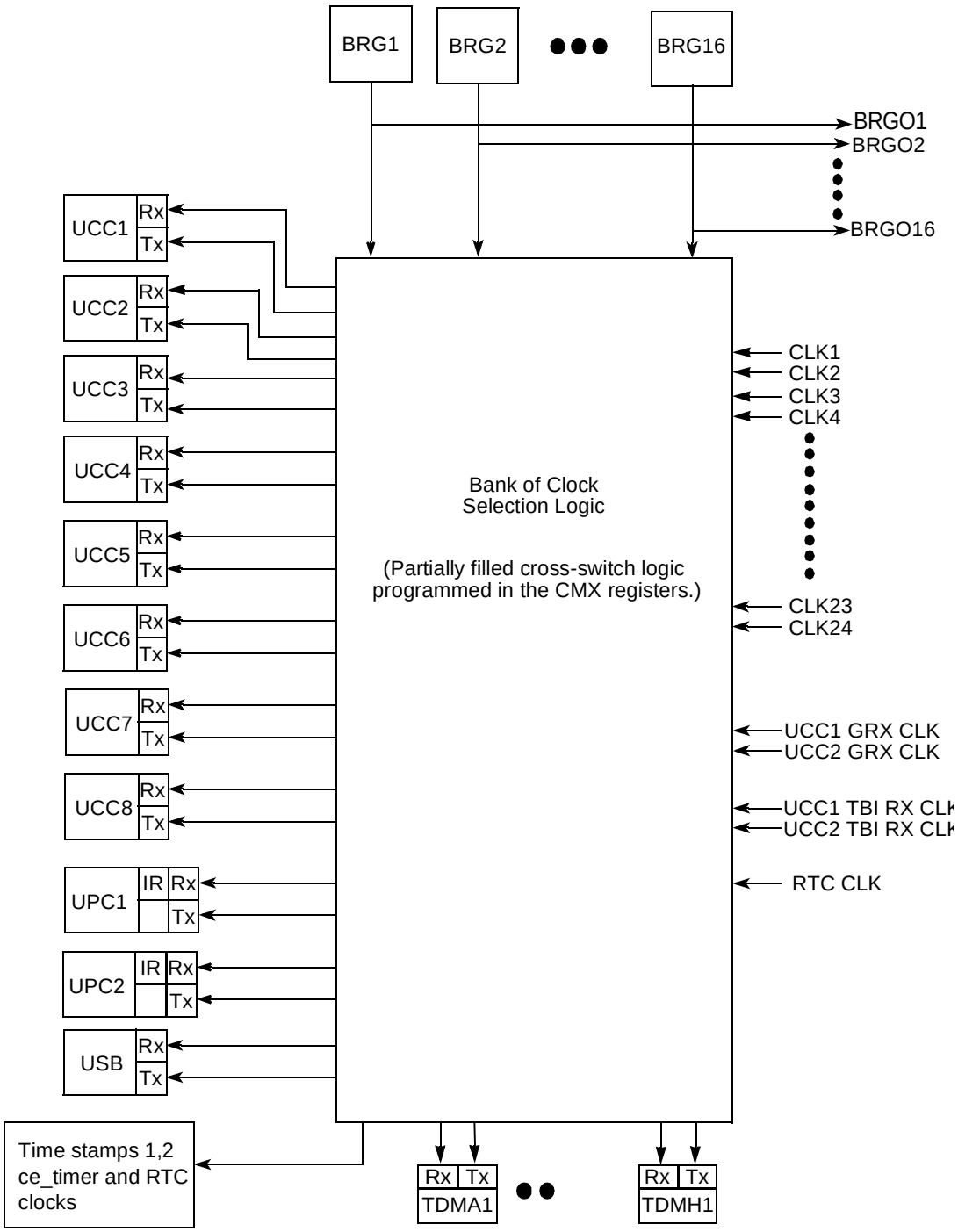


Figure 21-3. Bank of Clocks

The 16 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*x* pins may not always be available. Chapter 3, "Signal Descriptions," shows the function multiplexing.

Consult the tables in this chapter for restrictions in the bank-of-clocks mapping. Table 21-1 shows the clock source options for the serial controllers and TDM channels. Yellow coloring in this table stands for clocks that are common to at least 4 UCCs or TDMs.

Table 21-1. 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	20	21	22	23	24	RTC
UCC1 Rx[1]									V	V	V	V			V	V									
UCC1 Tx[2]									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	V					
UCC5 Tx													V	V	V	V			V	V					
UCC7 Rx													V	V	V	V			V	V					
UCC7 Tx													V	V	V	V			V	V					
UCC2 Rx[3]			V	V			V	V								V	V	V							
UCC2 Tx[4]			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							
UCC6 Rx					V	V	V	V								V					V	V			
UCC6 Tx					V	V	V	V								V					V	V			
UCC8 Rx					V	V	V	V								V					V	V			
UCC8 Tx					V	V	V	V								V					V	V			
UPC1 Rx													V	V	V	V			V	V					
UPC1 Tx													V	V	V	V			V	V					
UPC2 Rx					V	V	V	V											V	V					
UPC2 Tx					V	V	V	V											V	V					
UPC1 IR																	V	V	V						
UPC2 IR																		V	V	V					
USB			V		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										
TDME1 Rx											V					V							V	V	
TDME1 Tx												V					V						V	V	
TDMF1 Rx													V					V					V	V	
TDMF1 Tx														V					V				V	V	
TDMG1 Rx															V					V			V	V	
TDMG1 Tx																V					V		V	V	
TDMH1 Rx																	V					V	V	V	

Table 21-1. Clock Source Options—External Clock Signals (continued)

Clock	External CLK Number																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	RTC
TDMH1 Tx			V															V					V	V	
Time Stamp 1											V	V									V				V
Time Stamp 2											V	V									V				V
QE Timer											V	V									V				V
QE RTC							V	V							V	V									V

1 UCC1 Rx clock can be also derived from GRX clock.

2 UCC1Tx clock can be also derived from TBI RX CLK1. The UCC1 Tx clock is also the 125-MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI.

3 UCC2 Rx clock can be also derived from GRX clock.

4 UCC2 Tx clock can be also derived from TBI RX CLK1. The UCC2 Tx clock is also the 125-MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI.

Table 21-2. Clock Source Options—Internal Clock Generators

Clock	BRG Clock Number															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	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								
UCC7 Rx					V	V	V	V								
UCC7 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
UCC6 Rx													V	V	V	V
UCC6 Tx													V	V	V	V
UCC8 Rx													V	V	V	V
UCC8 Tx													V	V	V	V
UPC1 Rx					V	V	V	V								
UPC1 Tx					V	V	V	V								
UPC2 Rx													V	V	V	V
UPC2 Tx													V	V	V	V
UPC1 IR			V	V												
UPC2 IR			V	V												
USB									V	V						

Table 21-2. Clock Source Options—Internal Clock Generators (continued)

Clock	BRG Clock Number															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
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												
TDME1 Rx												V	V			
TDME1 Tx												V	V			
TDMF1 Rx												V	V			
TDMF1 Tx												V	V			
TDMG1 Rx												V	V			
TDMG1 Tx												V	V			
TDMH1 Rx												V	V			
TDMH1 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					
TDME1 Rsync													V	V		
TDME1 Tsync													V	V		
TDMF1 Rsync													V	V		
TDMF1 Tsync													V	V		
TDMG1 Rsync													V		V	
TDMG1 Tsync													V		V	
TDMH1 Rsync													V		V	
TDMH1 Tsync													V		V	
Time Stamp 1											V					
Time Stamp 2											V					
QUICC Engine Timer											V					
QE RTC			V	V												

After a clock source is selected, the clock is given an internal name. For the UCCs the names are RCLK*x* and TCLK*x*. These internal names specify the clocks sent to the UCCs and are used only in NMSI mode.

Table 21-3. 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	12	13	14	15	16
UCC1 Rx	V															
UCC1 Tx	V															
UCC3 Rx		V														
UCC3 Tx		V														
UCC5 Rx					V											
UCC5 Tx					V											
UCC7 Rx						V										
UCC7 Tx						V										
UCC2 Rx									V							
UCC2 Tx									V							
UCC4 Rx										V						
UCC4 Tx										V						
UCC6 Rx													V			
UCC6 Tx													V			
UCC8 Rx														V		
UCC8 Tx														V		

21.5 CMX Registers

The following sections describe the CMX registers.

Table 21-4. QUICC Engine MUX Register Summary

Address[1]	Name	Description
0x400	CMXGCR	Time stamps, SMI, USB
0x404	CMXSI1CRL	TDMA - TDMD RClk and TClk
0x408	CMXSI1CRH	TDME- TDMH RClk and TClk
0x40C	CMXSI1SYR	TDMA - TDMH TSYNC and RSYNC
0x410	CMXUCR1	UCC1, UCC3 RxClk and TxClk
0x414	CMXUCR2	UCC5, UCC7 RxClk and TxClk
0x418	CMXUCR3	UCC2, UCC4 RxClk and TxClk
0x41C	CMXUCR4	UCC6, UCC8 RxClk and TxClk
0x420	CMXUPCR	UPC1, UPC2 RxClk, TxClk and internal rate clocks

1 Offset from QUICC Engine Base address.

21.5.1 CMX General Clock Route Register (CMXGCR)

The CMX general clock route register (CMXGCR), seen in Figure 21-4, defines the Time stamps, Timer, USB clock sources and the SMI master.

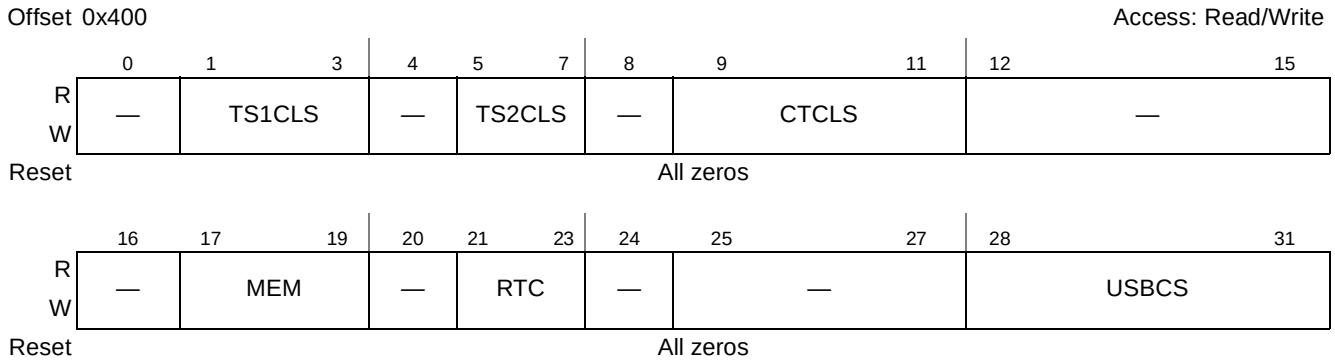


Figure 21-4. CMX General Clock Route Register (CMXGCR)

Table 21-5 describes CMXGCR fields.

Table 21-5. 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 Time Stamp 1 clock source is CLK21 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 Time Stamp 2 clock source is CLK21 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 Time Stamp 2 clock source is CLK21 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 21-5. 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. 000 UCC1 is the master of the SMI interface 001 UCC2 is the master of the SMI interface. 010 UCC3 is the master of the SMI interface. 011 UCC4 is the master of the SMI interface. 100 UCC5 is the master of the SMI interface. 101 UCC6 is the master of the SMI interface. 110 UCC7 is the master of the SMI interface. 111 UCC8 is the master of the SMI interface
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	—	Reserved. Should be cleared
25–27	—	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 USB clock is CLK21. 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 20.3.9, “QUICC Engine Time-Stamp Control Register (CETSCR),” and Section 20.3.8, “QUICC Engine Controller Configuration Register (CECCR).”

21.5.2 CMX SI1 Clock Route Low Register (CMXSI1CRL)

The CMX SI1 clock route low register (CMXSI1CRL), seen in Figure 21-5, 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	—	RTA1CS	—	RTB1CS	—	RTC1CS	—	RTD1CS
W								
Reset	All zeros							

	16	17–19	20	21–23	24	25–27	28	29–31
R	—	TTA1CS	—	TTB1CS	—	TTC1CS	—	TTD1CS
W								
Reset	All zeros							

Figure 21-5. CMX Si1 Clock Route Low Register (CMXSI1CRL)

Table 21-6 describes CMXSI1CRL fields.

Table 21-6. 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 common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM A1 receive clock is CLK2. (CLK2 can be programmed as 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 common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM B1 receive clock is CLK2. (CLK2 can be programmed as 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 21-6. 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 common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM C1 receive clock is CLK2. (CLK2 can be programmed as 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 common clock for TDM A1,B1,C1,D1 receive clock) 101 TDM D1 receive clock is CLK2. (CLK2 can be programmed as 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 common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM A1 transmit clock is CLK2. (CLK2 can be programmed as 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 common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM B1 transmit clock is CLK2. (CLK2 can be programmed as 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 21-6. 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 common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM C1 transmit clock is CLK2. (CLK2 can be programmed as 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 common clock for TDM A1,B1,C1,D1 transmit clock) 101 TDM D1 transmit clock is CLK2. (CLK2 can be programmed as common clock for TDM A1,B1,C1,D1 transmit clock) 110 TDM D1 transmit clock is CLK10. 111 TDM D1 transmit clock is CLK15.

21.5.3 CMX SI1 Clock Route High Register (CMXSI1CRH)

The CMX SI1 clock route high register (CMXSI1CRH), seen in Figure 21-6, defines the connection of SI1 TDM E-H to the clock sources that can be input from the bank of clocks.

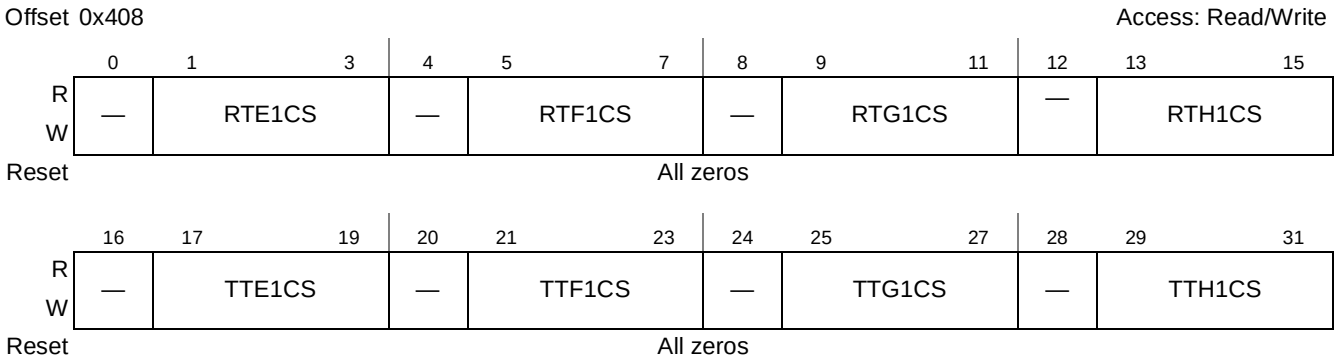


Figure 21-6. CMX Si1 Clock Route High Register (CMXSI1CRH)

Table 21-7 describes CMXSI1CRH fields.

Table 21-7. CMXSI1CRH Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1–3	RTE1CS	Receive TDM E1 clock source 000 TDM E1 receive clock is disabled. 001 TDM E1 receive clock is BRG12. 010 TDM E1 receive clock is BRG13. 011 Reserved 100 TDM E1 receive clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 101 TDM E1 receive clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 110 TDM E1 receive clock is CLK11. 111 TDM E1 receive clock is CLK16.
4	—	Reserved, should be cleared
5–7	RTF1CS	Receive TDM F1 clock source 000 TDM F1 receive clock is disabled. 001 TDM F1 receive clock is BRG12. 010 TDM F1 receive clock is BRG13. 011 Reserved 100 TDM F1 receive clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 101 TDM F1 receive clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 110 TDM F1 receive clock is CLK13. 111 TDM F1 receive clock is CLK18.
8	—	Reserved, should be cleared
9–11	RTG1CS	Receive TDM G1 clock source 000 TDM G1 receive clock is disabled. 001 TDM G1 receive clock is BRG12. 010 TDM G1 receive clock is BRG13. 011 Reserved 100 TDM G1 receive clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 101 TDM G1 receive clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 110 TDM G1 receive clock is CLK15. 111 TDM G1 receive clock is CLK20.
12	—	Reserved, should be cleared.

Table 21-7. CMXSI1CRH Field Descriptions (continued)

Bits	Name	Description
13–15	RTH1CS	Receive TDM H1 clock source 000 TDM H1 receive clock is disabled. 001 TDM H1 receive clock is BRG12. 010 TDM H1 receive clock is BRG13. 011 Reserved 100 TDM H1 receive clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 101 TDM H1 receive clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 receive clock) 110 TDM H1 receive clock is CLK17. 111 TDM H1 receive clock is CLK22.
16	—	Reserved, should be cleared
17–19	TTE1CS	Transmit TDM E1 clock source 000 TDM E1 transmit clock is disabled. 001 TDM E1 transmit clock is BRG12. 010 TDM E1 transmit clock is BRG13. 011 Reserved 100 TDM E1 transmit clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 101 TDM E1 transmit clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 110 TDM E1 transmit clock is CLK12. 111 TDM E1 transmit clock is CLK17.
20	—	Reserved, should be cleared
21–23	TTF1CS	Transmit TDM F1 clock source 000 TDM F1 transmit clock is disabled. 001 TDM F1 transmit clock is BRG12. 010 TDM F1 transmit clock is BRG13. 011 Reserved 100 TDM F1 transmit clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 101 TDM F1 transmit clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 110 TDM F1 transmit clock is CLK14. 111 TDM F1 transmit clock is CLK19.
24	—	Reserved, should be cleared
25–27	TTG1CS	Transmit TDM G1 clock source 000 TDM G1 transmit clock is disabled. 001 TDM G1 transmit clock is BRG12. 010 TDM G1 transmit clock is BRG13. 011 Reserved 100 TDM G1 transmit clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 101 TDM G1 transmit clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 110 TDM G1 transmit clock is CLK16. 111 TDM G1 transmit clock is CLK21.

Table 21-7. CMXSI1CRH Field Descriptions (continued)

Bits	Name	Description
28	—	Reserved, should be cleared
29–31	TTH1CS	Transmit TDM H1 clock source 000 TDM H1 transmit clock is disabled. 001 TDM H1 transmit clock is BRG12. 010 TDM H1 transmit clock is BRG13. 011 Reserved 100 TDM H1 transmit clock is CLK23. (CLK23 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 101 TDM H1 transmit clock is CLK24. (CLK24 can be programmed as common clock for TDM E1,F1,G1,H1 transmit clock) 110 TDM H1 transmit clock is CLK18. 111 TDM H1 transmit clock is CLK3.

21.5.4 CMX SI1 SYNC Route Register (CMXSI1SYR)

The CMX SI1 sync. route register (CMXSI1SYR), seen in Figure 21-7, defines the connection of SI1 to the sync. sources that can be input from the bank of clocks.

Offset 0x40C Access: Read/Write

	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
R W	RTA1SS	RTB1SS	RTC1SS	RTD1SS	RTE1SS	RTF1SS	RTG1SS	RTH1SS
Reset	All zeros							

	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
R W	TTA1SS	TTB1SS	TTC1SS	TTD1SS	TTE1SS	TTF1SS	TTG1SS	TTH1SS
Reset	All zeros							

Figure 21-7. CMX SI1 SYNC Route Register (CMXSI1SYR)

Table 21-8 describes CMXSI1SYR fields.

Table 21-8. 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

Table 21-8. CMXSI1SYR Field Descriptions (continued)

Bits	Name	Description
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
8–9	RTE1SS	Receive TDM E1 sync. source 00 TDM E1 receive sync is TDM_E1 RSYNC pin 01 TDM E1 receive sync is generated by BRG13. 10 TDM E1 receive sync is generated by BRG14. 11 Reserved
10–11	RTF1SS	Receive TDM F1 sync. source 00 TDM F1 receive sync is TDM_F1 RSYNC pin 01 TDM F1 receive sync is generated by BRG13. 10 TDM F1 receive sync is generated by BRG14. 11 Reserved
12–13	RTG1SS	Receive TDM G1 sync. source 00 TDM G1 receive sync is TDM_G1 RSYNC pin 01 TDM G1 receive sync is generated by BRG13. 10 TDM G1 receive sync is generated by BRG15. 11 Reserved
14–15	RTH1SS	Receive TDM H1 sync. source 00 TDM H1 receive sync is TDM_H1 RSYNC pin 01 TDM H1 receive sync is generated by BRG13. 10 TDM H1 receive sync is generated by BRG15. 11 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

Table 21-8. CMXSI1SYR Field Descriptions (continued)

Bits	Name	Description
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
24–25	TTE1SS	transmit TDM E1 sync. source 00 TDM E1 transmit sync is TDM_E1 TSYNC pin 01 TDM E1 transmit sync is generated by BRG13. 10 TDM E1 transmit sync is generated by BRG14. 11 Reserved
26–27	TTF1SS	transmit TDM F1 sync. source 00 TDM F1 transmit sync is TDM_F1 TSYNC pin 01 TDM F1 transmit sync is generated by BRG13. 10 TDM F1 transmit sync is generated by BRG14. 11 Reserved
28–29	TTG1SS	transmit TDM G1 sync. source 00 TDM G1 transmit sync is TDM_G1 TSYNC pin 01 TDM G1 transmit sync is generated by BRG13. 10 TDM G1 transmit sync is generated by BRG15. 11 Reserved
30–31	TTH1SS	transmit TDM H1 sync. source 00 TDM H1 transmit sync is TDM_H1 TSYNC pin 01 TDM H1 transmit sync is generated by BRG13. 10 TDM H1 transmit sync is generated by BRG15. 11 Reserved

21.5.5 CMX UCC Clock Route Register (CMXUCR1)

The CMX UCC clock route register (CMXUCR1), shown in Figure 21-8, defines the connection of the UCCs 1,3 to the TSA and to the clock sources from the bank of clocks.

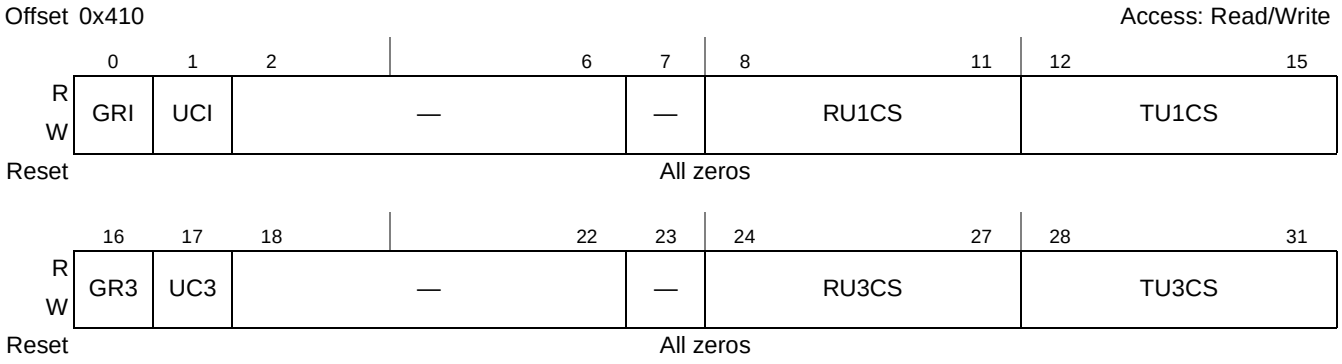


Figure 21-8. CMX UCC Clock Route Register (CMXUCR1)

Table 21-9 describes CMXUCR1 fields.

Table 21-9. 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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "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 NMSIx 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 NMSIx pins are available for other purposes.
2–6	—	Reserved, should be cleared
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 UCC1,3,5,7 receive and transmit clocks) 1010 UCC1 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011 UCC1 receive clock is GRX_CLK (use for soft switch from xGMII to MII for auto-negotiation) 1100–1111 Reserved
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

Table 21-9. CMXUCR1 Field Descriptions (continued)

Bits	Name	Description
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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "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 NMSIx 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 NMSIx pins are available for other purposes.
18–22	—	Reserved, should be cleared
23	—	Reserved, should be cleared
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 UCC1,3,5,7 receive and transmit clocks) 1010 UCC3 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 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 UCC1,3,5,7 receive and transmit clocks) 1010 UCC3 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved

21.5.6 CMX UCC Clock Route Register (CMXUCR2)

The CMX UCC clock route register (CMXUCR2), seen in Figure 21-9, defines the connection of the UCCs 5,7 to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

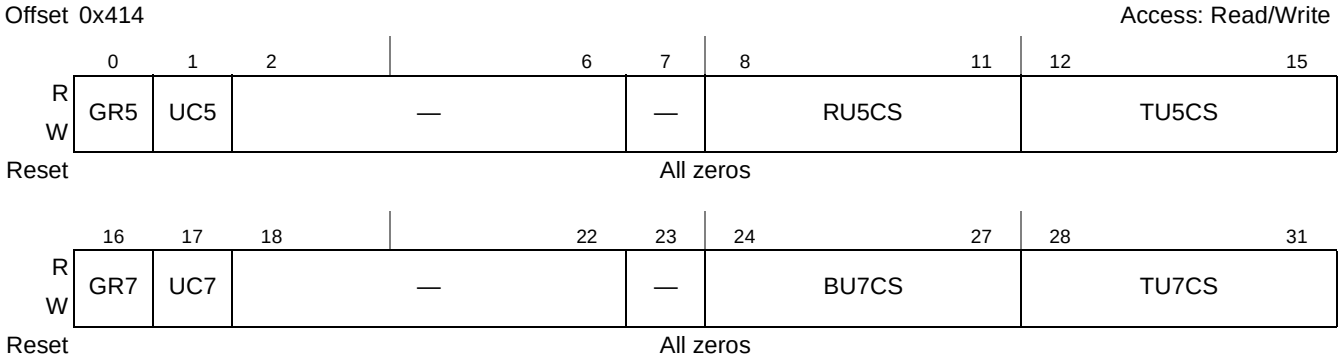


Figure 21-9. CMX UCC Clock Route Register (CMXUCR2)

Table 21-10 describes CMXUCR2 fields.

Table 21-10. 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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "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 NMSIx 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 NMSIx pins are available for other purposes.
2–6	—	Reserved, should be cleared
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 UCC5 receive clock is CLK20. 1001 UCC5 receive clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC5 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved

Table 21-10. 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 UCC5 transmit clock is CLK20. 1001 UCC5 transmit clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC5 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved
16	GR7	$\overline{\text{CTS}}$ mode of UCC7 (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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
17	UC7	Defines the UCC7 connection. 0 UCC7 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC7 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
18–22	—	Reserved, should be cleared
23	—	Reserved, should be cleared

Table 21-10. CMXUCR2 Field Descriptions (continued)

Bits	Name	Description
24–27	RU7CS	Receive UCC7 clock source (NMSI mode). Ignored if UCC7 is connected to the TSA (UC7 = 1). 0000 UCC7 receive clock is disabled. 0001 UCC7 receive clock is BRG5. 0010 UCC7 receive clock is BRG6. 0011 UCC7 receive clock is BRG7. 0100 UCC7 receive clock is BRG8. 0101 UCC7 receive clock is CLK13. 0110 UCC7 receive clock is CLK14. 0111 UCC7 receive clock is CLK19. 1000 UCC7 receive clock is CLK20. 1001 UCC7 receive clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC7 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved
28–31	TU7CS	Transmit UCC7 clock source (NMSI mode). Ignored if UCC7 is connected to the TSA (UC7 = 1). 0000 UCC7 transmit clock is disabled. 0001 UCC7 transmit clock is BRG5. 0010 UCC7 transmit clock is BRG6. 0011 UCC7 transmit clock is BRG7. 0100 UCC7 transmit clock is BRG8. 0101 UCC7 transmit clock is CLK13. 0110 UCC7 transmit clock is CLK14. 0111 UCC7 transmit clock is CLK19. 1000 UCC7 transmit clock is CLK20. 1001 UCC7 transmit clock is CLK 15 (CLK15 can be programmed as common clock for UCC1,3,5,7 receive and transmit clocks) 1010 UCC7 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC1–8 receive and transmit clocks) 1011–1111 Reserved

21.5.7 CMX UCC Clock Route Register (CMXUCR3)

The CMX UCC clock route register (CMXUCR3), shown in Figure 21-10, defines the connection of the UCCs 2, 4 to the TSA and to the clock sources from the bank of clocks.

Offset 0x418 Access: Read/Write

	0	1	2	6	7	8	11	12	15
R W	GR2	UC2	—		—	RU2CS		TU2CS	
Reset	All zeros								

	16	17	18	22	23	24	27	28	31
R W	GR4	UC4	—		—	RU4CS		TU4CS	
Reset	All zeros								

Figure 21-10. CMX UCC Clock Route Register (CMXUCR3)

Table 21-11 describes CMXUCR3 fields.

Table 21-11. 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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "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 NMSIx 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 NMSIx pins are available for other purposes.
2–6	—	Reserved, should be cleared
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 common clock for UCC2,4,6,8 receive and transmit clocks) 1010 UCC2 receive clock is CLK 8 (CLK8 can be programmed as common clock for UCC2,4,6,8 receive and transmit clocks) 1011 UCC2 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1100 UCC2 receive clock is GRX_CLK 1101–1111 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 common clock for UCC2,4,6,8 transmit and transmit clocks) 1010 UCC2 transmit clock is CLK 8 (CLK8 can be programmed as common clock for UCC2,4,6,8 transmit and transmit clocks) 1011 UCC2 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1100 UCC2 transmit clock is TBI RX CLK1 1101–1111 Reserved

Table 21-11. 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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "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 NMSIx 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 NMSIx pins are available for other purposes.
18–22	—	Reserved, should be cleared
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 common clock for UCC2,4,6,8 receive and transmit clocks) 1010 UCC4 receive clock is CLK 8 (CLK8 can be programmed as common clock for UCC2,4,6,8 receive and transmit clocks) 1011 UCC4 receive clock is CLK 16 (CLK16 can be programmed as 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 common clock for UCC2,4,6,8 transmit and transmit clocks) 1010 UCC4 transmit clock is CLK 8 (CLK8 can be programmed as common clock for UCC2,4,6,8 transmit and transmit clocks) 1011 UCC4 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved

21.5.8 CMX UCC Clock Route Register (CMXUCR4)

The CMX UCC clock route register (CMXUCR4), seen in Figure 21-11, defines the connection of the UCCs 6,8 to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

Offset 0x41C Access: Read/Write

	0	1	2–6	7	8–11	12–15
R W	GR6	UC6	—	—	RU6CS	TU6CS
Reset	All zeros					

	16	17	18–22	23	24–27	28–31
R W	GR8	UC8	—	—	RU8CS	TU8CS
Reset	All zeros					

Figure 21-11. CMX UCC Clock Route Register (CMXUCR4)

Table 21-12 describes CMXUCR4 fields.

Table 21-12. CMXUCR4 Field Descriptions

Bits	Name	Description
0	GR6	$\overline{\text{CTS}}$ mode of UCC6 (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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
1	UC6	Defines the UCC6 connection 0 UCC6 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC6 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
2–6	—	Reserved, should be cleared
7	—	Reserved, should be cleared

Table 21-12. CMXUCR4 Field Descriptions (continued)

Bits	Name	Description
8–11	RU6CS	Receive UCC6 clock source (NMSI mode). Ignored if UCC6 is connected to the TSA (UC6 = 1). 0000 UCC6 receive clock is disabled. 0001 UCC6 receive clock is BRG13. 0010 UCC6 receive clock is BRG14. 0011 UCC6 receive clock is BRG15. 0100 UCC6 receive clock is BRG16. 0101 UCC6 receive clock is CLK5. 0110 UCC6 receive clock is CLK6. 0111 UCC6 receive clock is CLK21. 1000 UCC6 receive clock is CLK22. 1001 UCC6 receive clock is CLK 7 (CLK7 can be programmed as common clock for UCC5–8 receive and transmit clocks) 1010 UCC6 receive clock is CLK 8 (CLK8 can be programmed as common clock for UCC5–8 receive and transmit clocks) transmit and transmit clocks) 1011 UCC6 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1110–1111 Reserved
12–15	TU6CS	Transmit UCC6 clock source (NMSI mode). Ignored if UCC6 is connected to the TSA (UC6 = 1). 0000 UCC6 transmit clock is disabled. 0001 UCC6 transmit clock is BRG13. 0010 UCC6 transmit clock is BRG14. 0011 UCC6 transmit clock is BRG15. 0100 UCC6 transmit clock is BRG16. 0101 UCC6 transmit clock is CLK5. 0110 UCC6 transmit clock is CLK6. 0111 UCC6 transmit clock is CLK21. 1000 UCC6 transmit clock is CLK22. 1001 UCC6 transmit clock is CLK 7 (CLK7 can be programmed as common clock for UCC5–8 receive and transmit clocks) 1010 UCC6 transmit clock is CLK 8 (CLK8 can be programmed as common clock for UCC5–8 receive and transmit clocks) transmit and transmit clocks) 1011 UCC6 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1110–1111 Reserved
16	GR8	$\overline{\text{CTS}}$ mode of UCC8 (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 GMx bit of the TDM's Mode Register (SIxMR1), see Section 36.6.4, "SI Mode Register (SIxMR)," and clear CTSP. $\overline{\text{CTS}}$ is derived from the Grant signal.
17	UC8	Defines the UCC8 connection. 0 UCC8 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus UCCn pins is made in the parallel I/O control register. 1 UCC8 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
18–22	—	Reserved, should be cleared
23	—	Reserved, should be cleared

Table 21-12. CMXUCR4 Field Descriptions (continued)

Bits	Name	Description
24–27	RU8CS	Receive UCC8 clock source (NMSI mode). Ignored if UCC8 is connected to the TSA (UC8 = 1). 0000 UCC8 receive clock is disabled. 0001 UCC8 receive clock is BRG13. 0010 UCC8 receive clock is BRG14. 0011 UCC8 receive clock is BRG15. 0100 UCC8 receive clock is BRG16. 0101 UCC8 receive clock is CLK5. 0110 UCC8 receive clock is CLK6. 0111 UCC8 receive clock is CLK21. 1000 UCC8 receive clock is CLK22. 1001 UCC8 receive clock is CLK 7 (CLK7 can be programmed as common clock for UCC 2,4,6,8 receive and transmit clocks) 1010 UCC8 receive clock is CLK 8 (CLK8 can be programmed as common clock for UCC 2,4,6,8 receive and transmit clocks) 1011 UCC8 receive clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved
28–31	TU8CS	Transmit UCC8 clock source (NMSI mode). Ignored if UCC8 is connected to the TSA (UC7 = 1). 0000 UCC8 transmit clock is disabled. 0001 UCC8 transmit clock is BRG13. 0010 UCC8 transmit clock is BRG14. 0011 UCC8 transmit clock is BRG15. 0100 UCC8 transmit clock is BRG16. 0101 UCC8 transmit clock is CLK5. 0110 UCC8 transmit clock is CLK6. 0111 UCC8 transmit clock is CLK21. 1000 UCC8 transmit clock is CLK22. 1001 UCC8 transmit clock is CLK 7 (CLK7 can be programmed as common clock for UCC 2,4,6,8 receive and transmit clocks) 1010 UCC8 transmit clock is CLK 8 (CLK8 can be programmed as common clock for UCC 2,4,6,8 receive and transmit clocks) 1011 UCC8 transmit clock is CLK 16 (CLK16 can be programmed as common clock for UCC 1–8 receive and transmit clocks) 1100–1111 Reserved

21.5.9 CMX UPC Clock Route Register (CMXUPCR)

The CMX UPC Internal Rate clock route register (CMXUPCR), seen in Figure 21-12, defines the source for the UPCs 1–2 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 each of the UPC's internal rate prescalar dividers.

Offset 0x420 Access: Read/Write

	0–3	4–7	8–10	11	12	13–15
R W	UPC1RCS	UP1TCS	—	—	—	UPC1IRCS
Reset	All zeros					

	16–19	20–23	24–26	27	28	29–31
R W	UPC2RCS	UP2TCS	—		—	UPC2IRCS
Reset	All zeros					

Figure 21-12. CMX UPC Clock Route Register (CMXUPCR)

Table 21-13 describes CMXUPCR fields.

Table 21-13. 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 UPC1 receive clock is CLK20 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 UPC1 transmit clock is CLK20 1011–1111 Reserved
8–10	—	Reserved. Should be cleared.
11	—	Reserved. Should be cleared.
12	—	Reserved. Should be cleared.

Table 21-13. 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
16–19	UPC2RCS	UPC2 receive clock source 0000 UPC2 receive clock is disabled 0001 UPC2 receive clock is BRG13 0010 UPC2 receive clock is BRG14 0011 UPC2 receive clock is BRG15 0100 UPC2 receive clock is BRG16 0101 UPC2 receive clock is CLK5 0110 UPC2 receive clock is CLK6 0111 UPC2 receive clock is CLK7 1000 UPC2 receive clock is CLK8 1001 UPC2 receive clock is CLK19 1010 UPC2 receive clock is CLK20 1011–1111 Reserved
20–23	UP2TCS	UPC2 transmit clock source 0000 UPC2 transmit clock is disabled 0001 UPC2 transmit clock is BRG13 0010 UPC2 transmit clock is BRG14 0011 UPC2 transmit clock is BRG15 0100 UPC2 transmit clock is BRG16 0101 UPC2 transmit clock is CLK5 0110 UPC2 transmit clock is CLK6 0111 UPC2 transmit clock is CLK7 1000 UPC2 transmit clock is CLK8 1001 UPC2 transmit clock is CLK19 1010 UPC2 transmit clock is CLK20 1011–1111 Reserved
24–26	—	Reserved. Should be cleared.
27	—	Reserved. Should be cleared.
28	—	Reserved. Should be cleared.
29–31	UPC2IRCS	UPC2 Internal Rate clock source 000 UPC2 internal rate clock is the transmit clock of UPC2 001 UPC2 internal rate clock is BRG3 010 UPC2 internal rate clock is BRG4 011 UPC2 internal rate clock is CLK18 100 UPC2 internal rate clock is CLK19 101 UPC2 internal rate clock is CLK20 110–111 Reserved

21.6 Baud-Rate Generators (BRGs)

The CMX contains 16 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:

- Sixteen independent and identical BRGs
- Limited On-the-fly changes allowed (See Section 21.7.1, "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 21-13 shows the block diagram for a BRG.

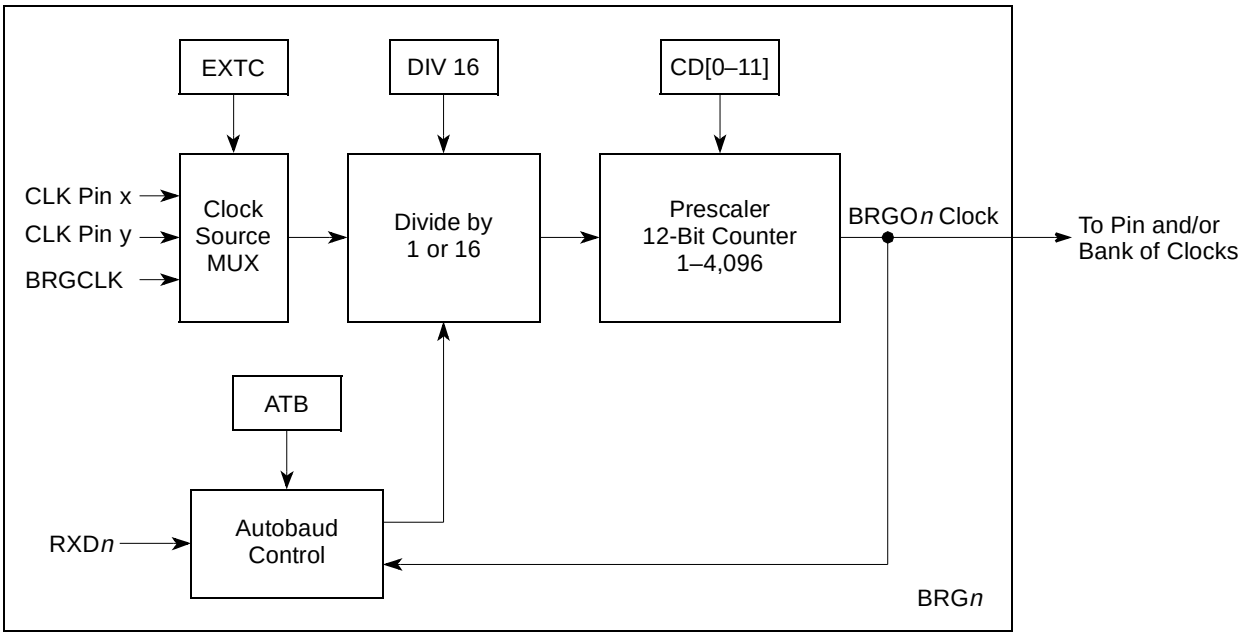


Figure 21-13. Baud-Rate Generator (BRG) Block Diagram

Each BRG clock source can be BRGCLK, or one of two external clocks (selected in BRGC*x*[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*x*[DIV16]) and a 12-bit prescaler (BRGC*x*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on-the-fly (excluding some specific division ratios – see Section 21.7.1, "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.

21.7 BRG Configuration Registers 1–16 (BRGC*x*)

The BRG configuration registers (BRGC*x*) are shown in Figure 21-14. 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 21.7.1, "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+0x2C (BRGC12), BRG_BASE+0x30 (BRGC13), BRG_BASE+0x34 (BRGC14), BRG_BASE+0x38 (BRGC15), BRG_BASE+0x3C (BRGC16) — Access: Read/Write

	0–13	14	15
R W	—	RST	EN
Reset	All zeros		

	16–17	18	19–30	31
R W	EXTC	ATB	CD	DIV16
Reset	All zeros			

Figure 21-14. Baud-Rate Generator Configuration Registers (BRGC*x*)

Table 21-14 describes the BRGC*x* fields.

Table 21-14. BRGC*x* Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	RST	Reset BRG. Performs a software reset of the BRG identical to that of an external reset. A reset disables the BRG and drives BRGO high. This is externally visible only if BRGO is connected to the corresponding parallel I/O pin. 0 Enable the BRG. 1 Reset the BRG (software reset).
15	EN	Enable BRG count. Used to dynamically stop the BRG from counting—useful for low-power modes. 0 Stop all clocks to the BRG. 1 Enable clocks to the BRG.

Table 21-14. BRGC*x* Field Descriptions (continued)

Bits	Name	Description
16–17	EXTC	External clock source. Selects the BRG input clock. See Table 21-15 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 20.3.8, "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, 13, 14: The BRG input clock comes from the CLK11 pin If BRG11, 12, 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 If BRG9, 10, 13, 14: The BRG input clock comes from the CLK21 pin If BRG11, 12, 15, 16: The BRG input clock comes from the CLK23 pin 11 Reserved.
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 25.3.7, "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 21.8, "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 21.9, "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 21.9, "UART Baud Rate Examples." 0 Divide by 1. 1 Divide by 16.

Table 21-15 shows the possible external clock sources for the BRGs.

Table 21-15. BRG External Clock Source Options

BRG	CLK																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
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										V			
BRG10											V										V			
BRG11													V										V	
BRG12													V										V	
BRG13											V										V			
BRG14											V										V			
BRG15													V										V	
BRG16													V										V	

21.7.1 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, unset EN, set RST and program the new value.
- When moving to or from Autobaud mode, issue the following sequence: Unset EN, set RST and program the new value.

21.8 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*x*[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 16 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*x*[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*x*. The interrupt handler can then adjust BRGC*x*[CD, DIV16] (see Table 21-16) 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.

21.8.1 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: unset EN, set RST and program the new value.

21.9 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 BRGCx[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 21-16 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 21-16. Typical Baud Rates for Asynchronous Communication

Baud Rate	Using a 66-MHz BRG Input Clock		
	BRGC*x*[DIV16]	BRGC*x*[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*x*[DIV16] = 0, and BRGC*x*[CD] = 389.

21.10 QUICC Engine Timers (GTM)

The global timer module in the QUICC Engine (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.

The programming model of the Timers is identical to the GTM of the system. The key difference is that the QUICC Engine timers are clocked by the QUICC Engine reference clock (half QE clock frequency) and do not have any I/O interface, therefore capture modes and output trigger are not available.

Figure 21-15 shows the functional GTM block diagram.

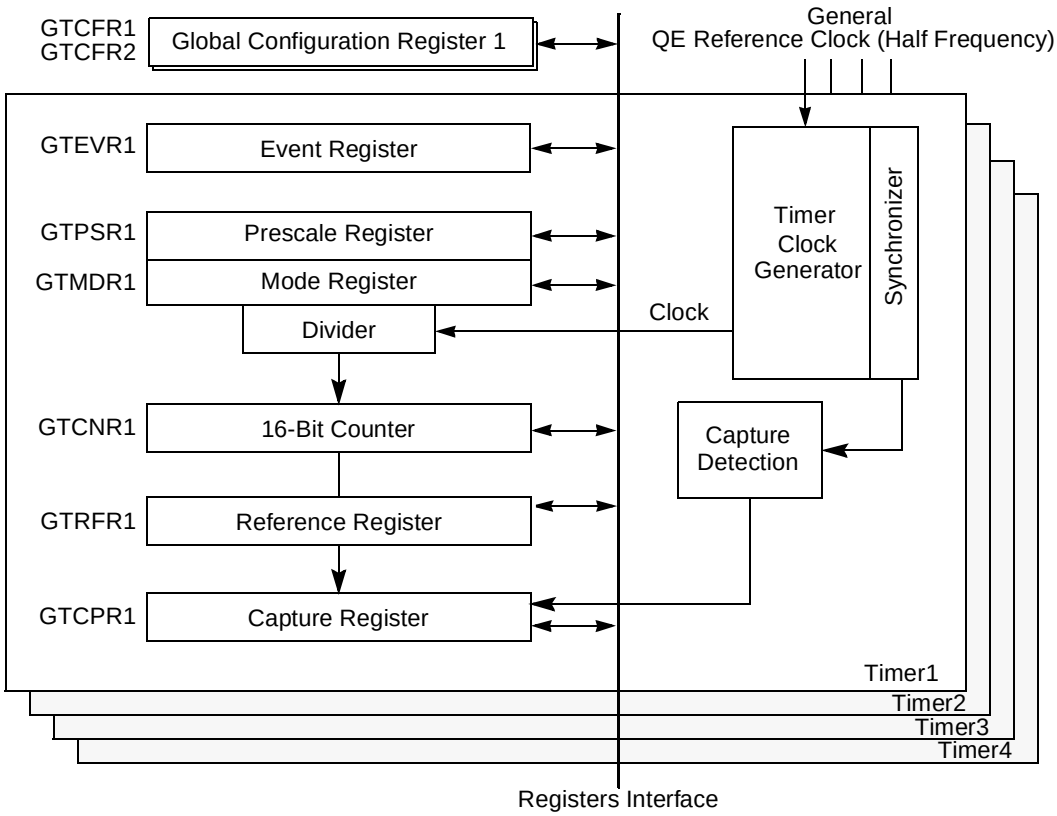


Figure 21-15. Global Timers Block Diagram

21.10.1 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

21.10.2 Modes of Operation

The GTM unit can operate in the following modes:

21.10.2.1 Cascaded Modes

GTCFRx[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. Since the decision to cascade timers is made independently, the user has the option of selecting two 16-bit timers and one 32-bit timer, 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.

21.10.2.2 Clock Source Modes

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

- The QUICC Engine reference clock (one half of the QE frequency)
- The system 'slow go' clock (QUICC Engine reference clock internally divided by 16)

21.10.2.3 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.

21.10.2.4 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 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.

21.10.3 External Signals

The QUICC Engine Timers do not have external signals.

21.10.4 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 Chapter 2, "Memory Map." Table 21-17 shows memory map of GTM.

Table 21-17. GTM Register Address Map

Offset	Register	Access	Reset Value	Section/ Page
0x00	Timer 1 and 2 global timers configuration register (GTCFR1)	R/W	0x00	21.10.4.1/21-41
0x01–0x03	Reserved	—	—	—
0x04	Timer 3 and 4 global timers configuration register (GTCFR2)	R/W	0x00	21.10.4.1/21-41
0x05–0x0F	Reserved	—	—	—
0x10	Timer 1 global timers mode register (GTMDR1)	R/W	0x0000	21.10.4.2/21-43
0x12	Timer 2 global timers mode register (GTMDR2)			
0x14	Timer 1 global timers reference register (GTRFR1)	R/W	0x0000	21.10.4.3/21-44
0x16	Timer 2 global timers reference register (GTRFR2)			
0x18	Timer 1 global timers capture register (GTCPR1)	R	0x0000	21.10.4.4/21-45
0x1A	Timer 2 global timers capture register (GTCPR2)			
0x1C	Timer 1 global timers counter register (GTCNR1)	R/W	0x0000	21.10.4.5/21-45
0x1E	Timer 2 global timers counter register (GTCNR2)			
0x20	Timer 3 global timers mode register (GTMDR3)	R/W	0x0000	21.10.4.2/21-43
0x22	Timer 4 global timers mode register (GTMDR4)			
0x24	Timer 3 global timers reference register (GTRFR3)	R/W	0x0000	21.10.4.3/21-44
0x26	Timer 4 global timers reference register (GTRFR4)			
0x28	Timer 3 global timers capture register (GTCPR3)	R	0x0000	21.10.4.4/21-45
0x2A	Timer 4 global timers capture register (GTCPR4)			
0x2C	Timer 3 global timers counter register (GTCNR3)	R/W	0x0000	21.10.4.5/21-45
0x2E	Timer 4 global timers counter register (GTCNR4)			

Table 21-17. GTM Register Address Map (continued)

Offset	Register	Access	Reset Value	Section/ Page
0x30	Timer 1 global timers event register (GTEVR1)	Special	0x0000	21.10.4.6/21-46
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	21.10.4.7/21-47
0x3A	Timer 2 global timers prescale register (GTPSR2)			
0x3C	Timer 3 global timers prescale register (GTPSR3)			
0x3E	Timer 4 global timers prescale register (GTPSR4)			

21.10.4.1 Global Timers Configuration Registers (GTCFR)

GTCFR1 and GTCFR2, shown in Figure 21-16 and Figure 21-17, 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, avoid changing the modes of operation and enabling the timer during the same register write operation. The mode can be changed when GTCFRn[RSTn] is cleared, but when GTCFRn[RSTn] 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 21-16. Global Timers Configuration Register 1 (GTCFR1)

Table 21-18 defines the bit fields of GTCFR1.

Table 21-18. 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 21-18.

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 21-17. Global Timers Configuration Register 2 (GTCFR2)

Table 21-19 defines the bit fields of GTCFR2.

Table 21-19. 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.

21.10.4.2 Global Timers Mode Registers (GTMDR1–GTMDR4)

GTMDR1, GTMDR2, GTMDR3, and GTMDR4 are shown in Figure 21-18. Erratic behavior may occur if GTCFR1 and GTCFR2 are not initialized before the GTMDRx. Only GTCFRx[RSTx] and GTCFRx[STPx] can be modified at any time.

Offset 0x10
0x12
0x20
0x22

Access: Read/Write

	0	7	8	9	10	11	12	13	14	15
R W	SPS		CE		OM	ORI	FRR	ICLK		GE
Reset	All zeros									

Figure 21-18. Global Timers Mode Registers (GTMDR1–GTMDR4)

Table 21-20 defines the bit fields of GTMDR.

Table 21-20. GTMDR 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.

21.10.4.3 Global Timers Reference Registers (GTRFR1–GTRFR4)

GTRFR1, GTRFR2, GTRFR3, and GTRFR4, shown in Figure 21-19 are a 16-bit, memory-mapped, read/write registers containing the 16-bit reference values for the each timer’s timeout. GTRFRx is set to all ones by reset. The reference value is not reached until GTCNRx[CNV] increments to value, equal GTRFRx[TRV].

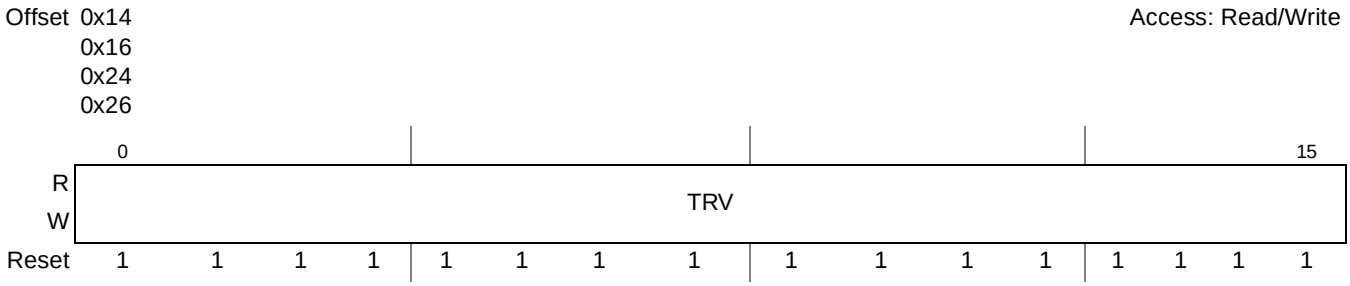


Figure 21-19. Global Timers Reference Registers (GTRFR1–GTRFR4)

Table 21-21 defines the bit fields of GTRFR.

Table 21-21. GTRFR 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.

21.10.4.4 Global Timers Capture Registers (GTCPR1–GTCPR4)

GTCPR1, GTCPR2, GTCPR3, and GTCPR4, shown in Figure 21-20, are non-functional read only registers.

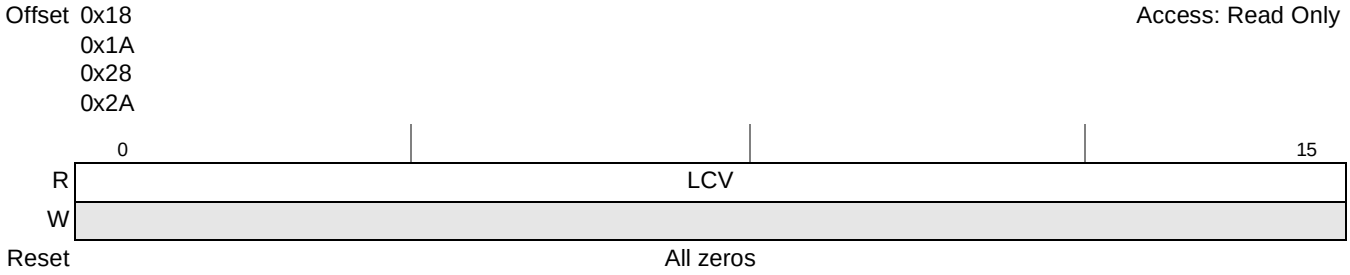


Figure 21-20. Global Timers Capture Registers (GTCPR1—GTCPR4)

Table 21-22 defines the bit fields of GTCPR.

Table 21-22. GTCPR Field Descriptions

Bits	Name	Description
0–15	LCV	Latched counter value Corresponding timer's 16-bit latched value.

21.10.4.5 Global Timers Counter Registers (GTCNR1–GTCNR4)

GTCNR1, GTCNR2, GTCNR3, and GTCNR4, shown in Figure 21-21, are four 16-bit, memory-mapped, read/write up-counters. A read cycle to a GTCNRx[CNV] fields yields the current value of the appropriate timer but does not affect the counting operation. A write cycle to a GTCNRx[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescaler counters to be reset.

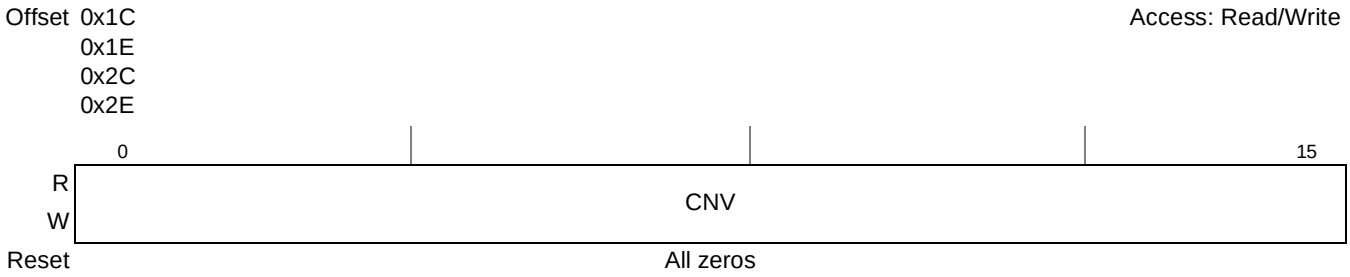


Figure 21-21. Global Timers Counter Registers (GTCNR1–GTCNR4)

Table 21-22 defines the bit fields of GTCNR.

Table 21-23. GTCNR Field Descriptions

Bits	Name	Description
0–15	CNV	Counter value. Corresponding timer's 16-bit read/write up-counter value.

21.10.4.6 Global Timers Event Registers (GTEVR1–GTEVR4)

GTEVR1, GTEVR2, GTEVR3, and GTEVR4, shown in Figure 21-22, are used to report events recognized by any of the timers. On recognition of an output reference event, the appropriate timer sets GTEVRx[REF], regardless of the corresponding GTMDRx[ORI] bit. The capture event is only set if it is enabled by GTMDRx[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.

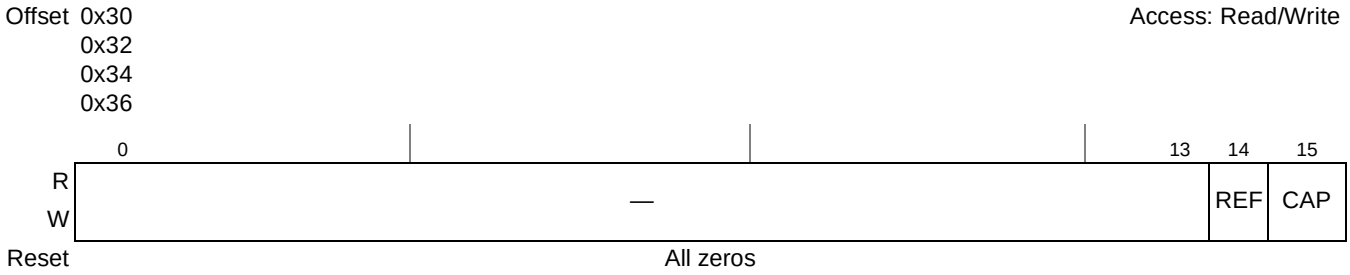


Figure 21-22. Global Timers Event Registers (GTEVR1–GTEVR4)

Table 21-24 defines the bit fields of GTEVR.

Table 21-24. 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 GTRFRx[TRV] value. GTMDRx[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Counter capture event: 0 No event 1 Reserved

21.10.4.7 Global Timers Prescale Registers (GTPSR1–GTPSR4)

GTPSR1, GTPSR2, GTPSR3, and GTPSR4 are shown in Figure 21-23. Erratic behavior may occur if GTPSRx is not initialized before the corresponding GTMDRx. 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 (GTPSRx[PPS] = 0x00, GTMDRx[SPS] = 0x00) to 65,536 (GTPSRx[PPS] = 0xFF, GTMDR[SPS] = 0xFF).

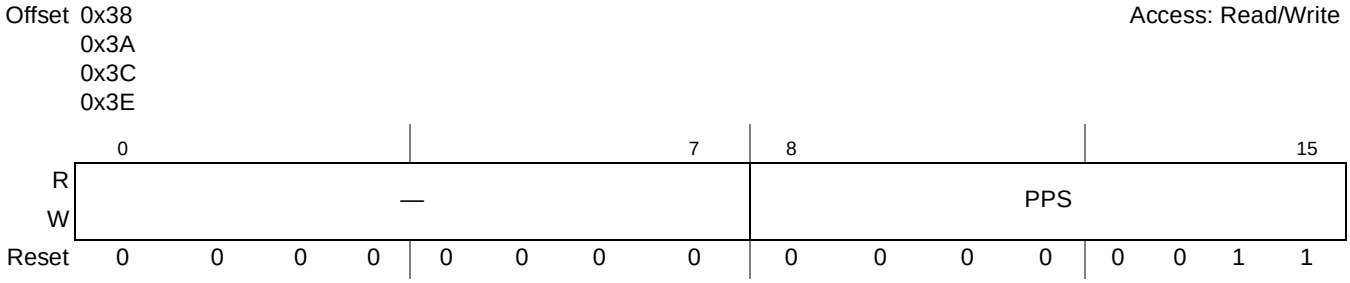


Figure 21-23. Global Timers Prescale Registers (GTPSR1–GTPSR4)

Table 21-25 defines the bit fields of GTPSR.

Table 21-25. 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.

21.10.5 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 TINx pin to be the clock source. TINx 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 GTMDRx[ICLK] bits. The prescalers (GTMDRx[SPS] and GTPSRx[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.

21.10.5.1 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 GTMRR selects each mode.

- Free run reference mode (GTMDRx[FRR]=0)

  The corresponding timer count continues to increment after the reference value is reached.
- Reset reference mode (GTMDRx[FRR]=1)

  The corresponding timer count is reset immediately after the reference value is reached.

Upon reaching the reference value, the corresponding GTEVRx[REF] bit is set and an interrupt is issued if GTMDRx[ORI] = 1.

21.10.5.2 Capture Modes

The capture modes are disabled in the QUICC Engine Timers.

21.10.5.3 Cascaded Modes

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

- Non-cascaded mode (GTCFRx[PCAS] = 0 and GTGCF2[SCAS] = 0)

  If GTCFRx[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 21-24). When working in the none-cascaded mode, the non-cascaded GTRFR, GTCPR, and GTCNR should be referenced with appropriate 16-bit bus cycles.

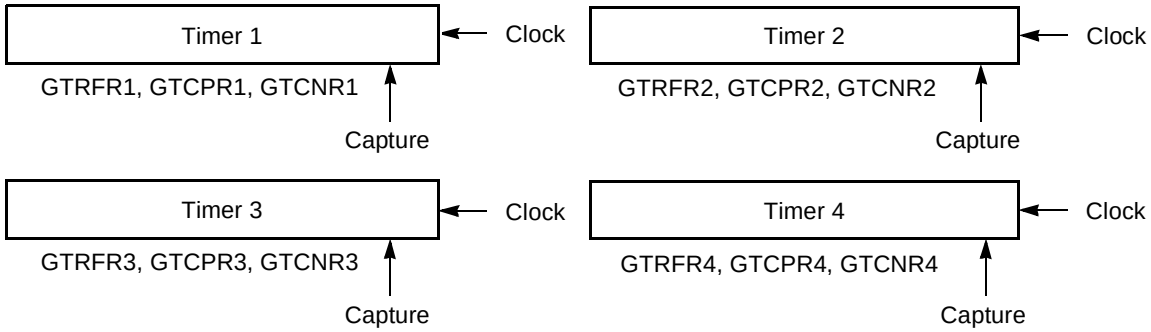


Figure 21-24. Timers Non-Cascaded Mode Block Diagram

- Pair-cascaded mode (GTCFR1[PCAS] = 1 and/or GTCFR2[PCAS] = 1, GTCFR2[SCAS] = 0). Two 16-bit timers are internally cascaded to form a 32-bit counter: timer 1 cam be internally cascaded to timer 2, and timer 3 can be internally cascaded to timer 4, as shown in Figure 21-25. 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[PCAS] = 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. GTMDR1/GTMDR3 is ignored, and the modes and functions are defined using GTMDR2/GTMDR4 and GTCFR1/GTCFR2. The captures are 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.

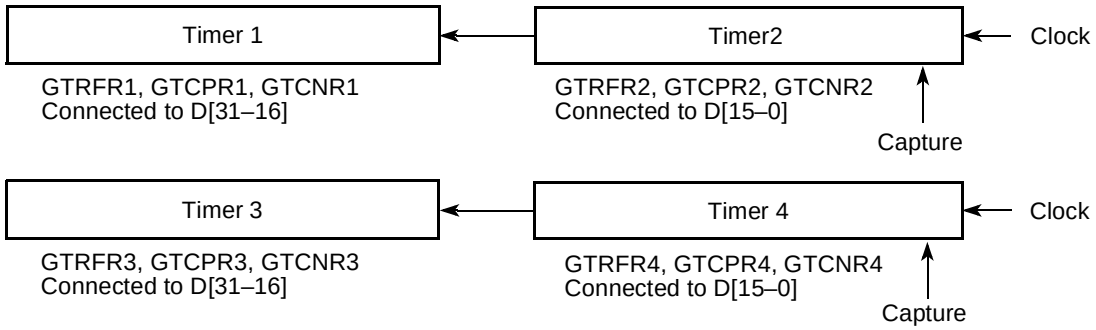


Figure 21-25. Timer Pair-Cascaded Mode Block Diagram

- Super-cascaded mode (GTCFR2[SCAS] = 1). All four 16-bit timers can be internally cascaded to form a 64-bit counter, as shown in Figure 21-26. If GTCFR2[SCAS] = 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 are 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.

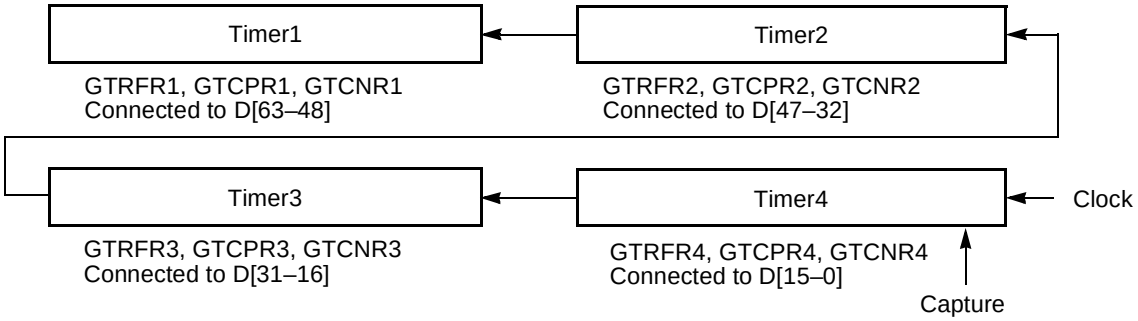


Figure 21-26. Timers Super-Cascaded Mode Block Diagram

21.10.6 Initialization/Applications Information

The following initialization sequence is recommended for the GTM registers:

1. Write to GTCFRx to reset, to stop or to configure the appropriate timer's operation: cascaded timers configuration, gate mode configuration.
2. Write to GTPSRx[PPS] fields to program the appropriate timer clock primary prescaler.
3. Write to GTMDRx 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 GTCNRx according to appropriate timer GTMDRx programming.

   A write cycle to a GTCNRx[CNV] fields sets the register to the written value, causing its corresponding primary and secondary prescalers, (GTPSRx[PPS] and GTMDRx[SPS]), to be reset.
7. Write to GTGCRx[STP] and to GTGCRx[RST] in order to initialize the appropriate timer's operation.

Chapter 22
Serial Peripheral Interface (SPI)

22.1 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 22-1, 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 22.1.3.4, “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.

22.1.1 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
- Communication with Ethernet PHY for configuration and status (MIIMCOM-MII management communication protocol)
- Multi-MIIMCOM environment with up to 32 PHYs
- Programmable clock gap between two characters in master mode
- Controlled by CPU/QUICC Engine RISC according to user configuration.

22.1.2 Block Diagram

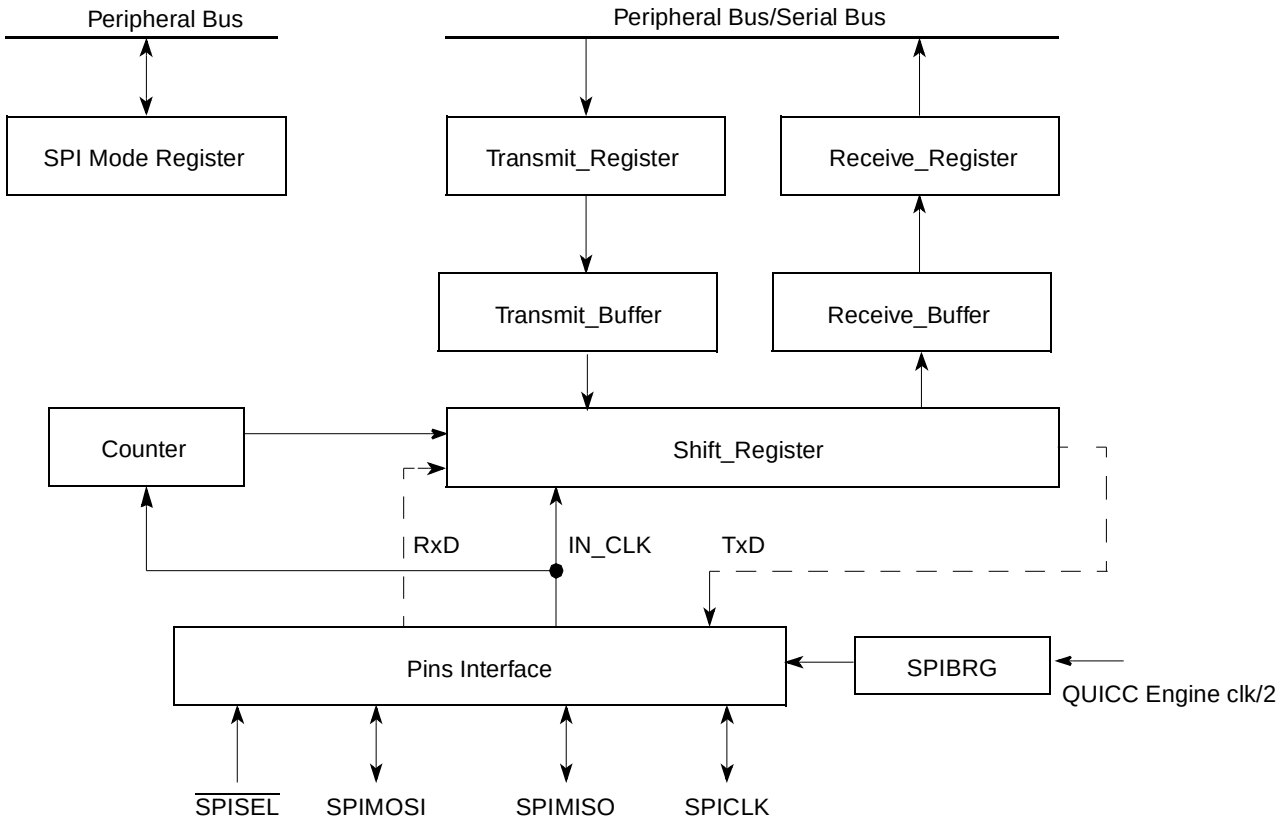


Figure 22-1. SPI Block Diagram

22.1.3 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.

22.1.3.1 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 22-2. 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.

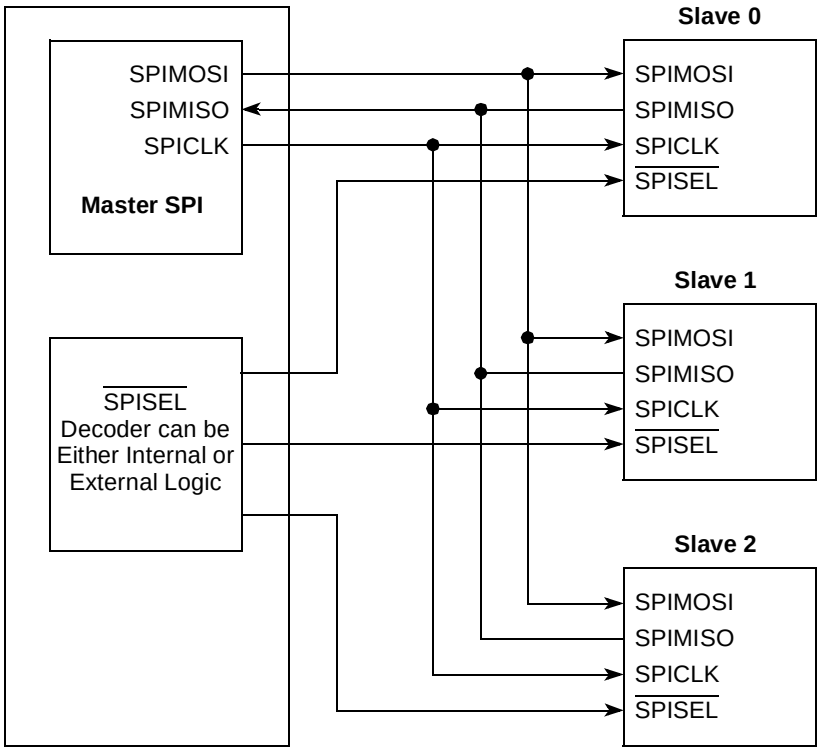


Figure 22-2. Single-Master/Multi-Slave Configuration

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.

22.1.3.2 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 CPU 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

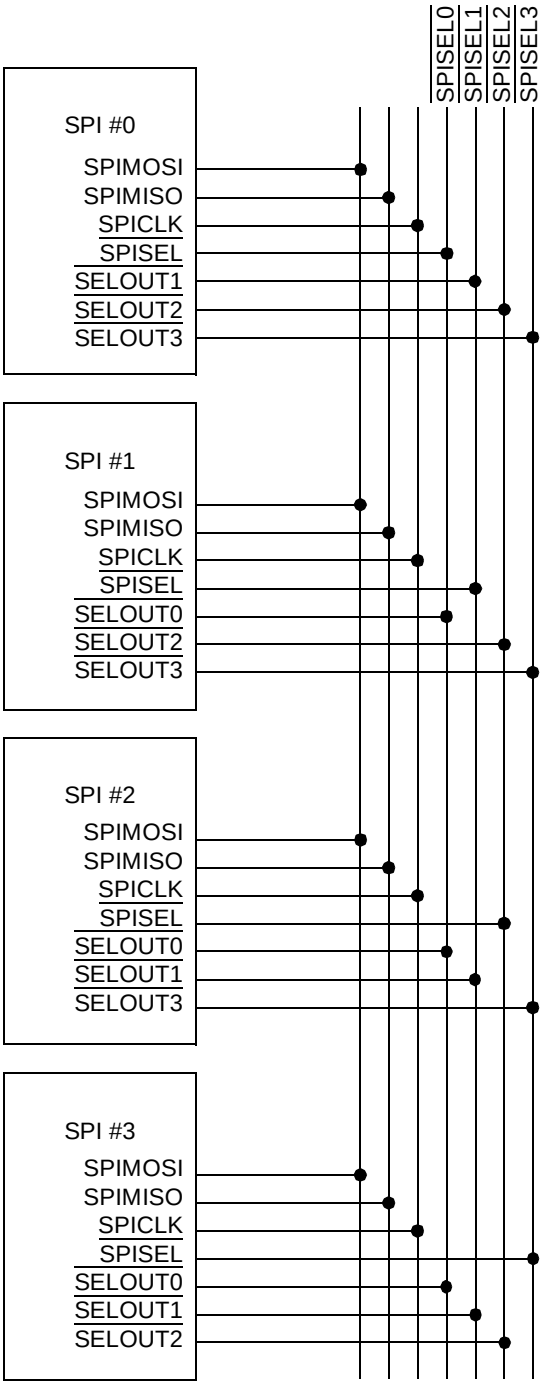
When enabling the SPI or changing parameters in SPI Mode Register (like CP,CI), $\overline{\text{SPISEL}}$ must remain negated for at least 2 QUICC Engine clk/2 clocks afterwards.

Also if $\overline{\text{SPISEL}}$ is negated between transfers, its negation time should be at least 2 QUICC Engine clk/2 clocks.

22.1.3.3 SPI in Multi-master Operation

The SPI can operate in a multi-master environment in which SPI devices are connected to the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK signals of all SPIs are shared; the $\overline{\text{SPISEL}}$ inputs are connected separately, as shown in Figure 22-3. 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 CPU 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 S 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 22-3. Multimaster Configuration

22.1.3.4 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 gigabit media independent interface (GMII) 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 22-4, Figure 22-5, and Figure 22-6 for more information.

22.1.3.4.1 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.

22.1.3.4.2 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 22-6) 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 22-6.

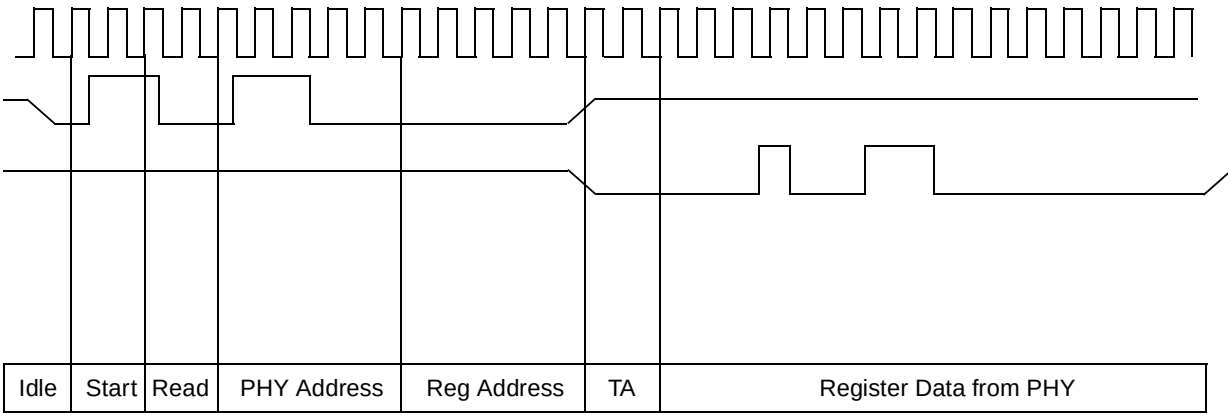


Figure 22-4. Typical Ethernet PHY Management Protocol for Read Operation

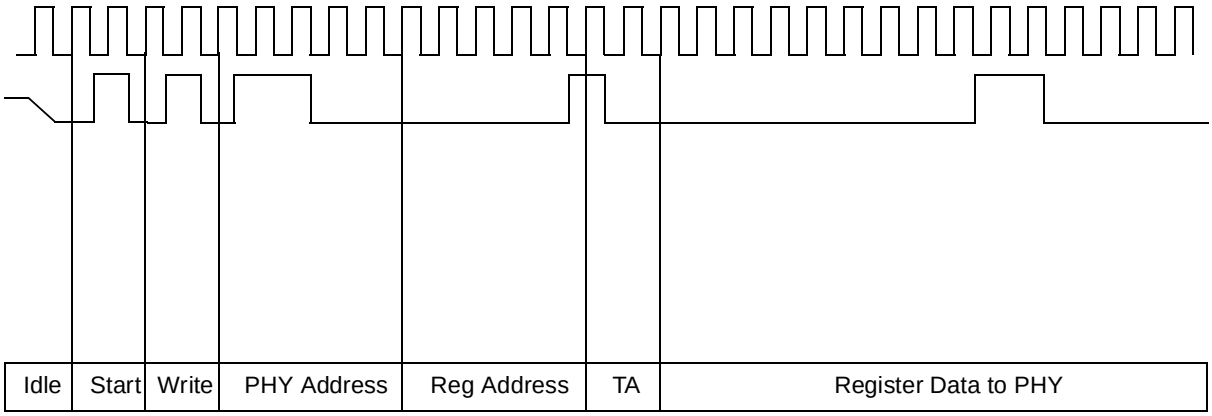


Figure 22-5. Typical Ethernet PHY Management Protocol for Write Operation

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 ddddddddddddddd

Figure 22-6. Serial Management Interface Protocol

Figure 22-7 describes the connectivity to Ethernet PHY.

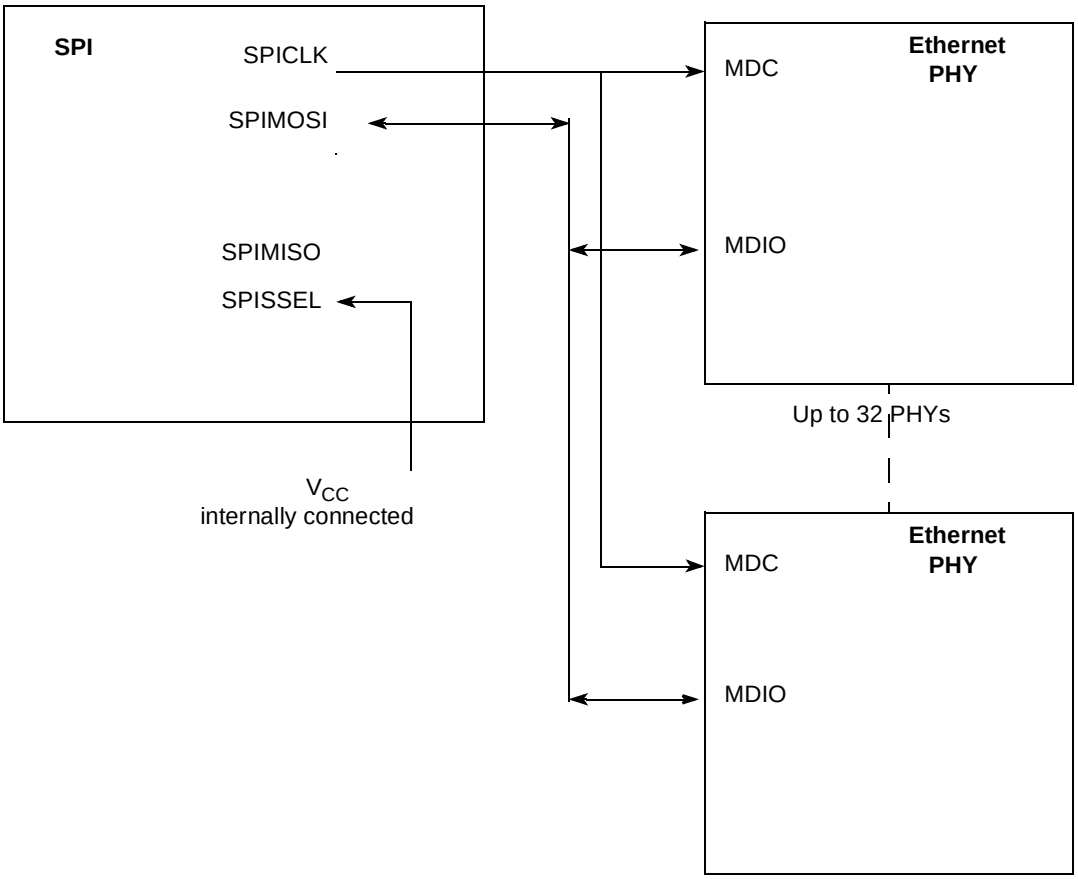


Figure 22-7. MIIMCOM Connectivity to Ethernet PHY

22.2 External Signal Descriptions

The SPI supports a four-wire interface—transmit, receive, clock, and slave select.

22.2.1 Overview

Table 22-1. 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

22.2.2 Detailed Signal Descriptions

Table 22-2. 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.

22.3 Programming the SPI Registers

Table 22-3. SPI Registers Summary in QUICC Engine Mode

Offset from CE_base	Register	Access	Reset Value
0x4E0/0x520	SPMODE–SPI mode register	R/W	0x0000_0000
0x4E6/0x526	SPIE–SPI event register	R/W	0x00
0x4EA/0x52A	SPIM–SPI mask register	R/W	0x00
0x4ED/0x52D	SPCOM–SPI command register	W	0x00

NOTE

0x4E0 is the offset for the first SPI in 836x platform. 0x520 is the offset for the second SPI in the 836x platform dedicated for MIIMCOM.

22.3.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 22-8, controls both the SPI operation mode and clock source.

Offset: 0x4E0/0x520 Access: Read/Write

	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									

	16	17	18	19	20–24	25–31
R W	—	OP	MII	—	CG	—
Reset	All zeros					

Figure 22-8. SPMODE-SPI Mode Register

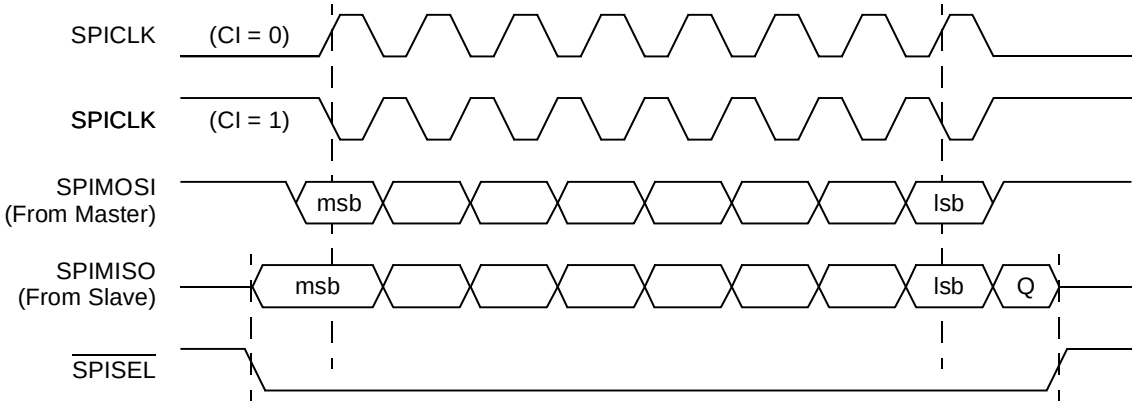
Table 22-4 describes the SPMODE fields.

Table 22-4. 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 22-9 and Figure 22-10. 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 22-9 and Figure 22-10. 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 22.3.1.1, “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 22-4. SPMODE Field Descriptions (continued)

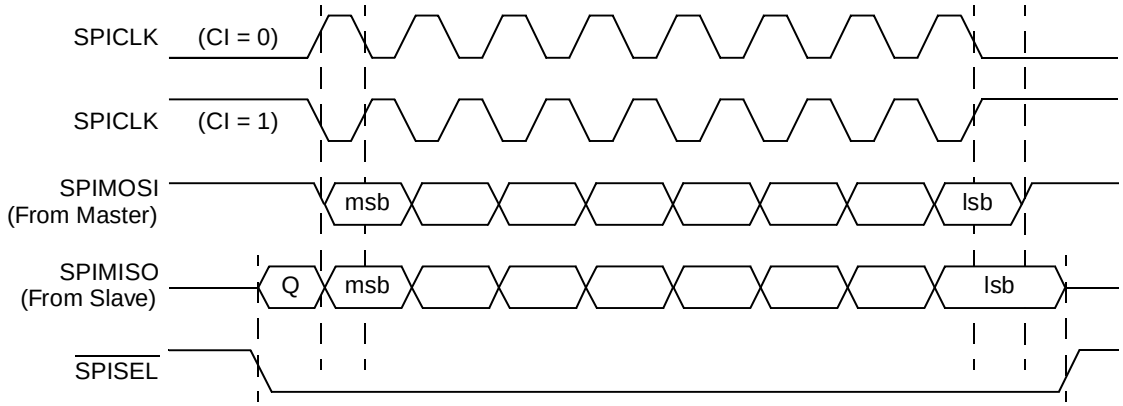
Bits	Name	Description
20–24	CG	Clock Gap For SPI Master mode: insert gaps between two transmitted characters according to CG size. Example: CG = 00101 inserts 5 bits time gap between every two consecutive characters For MIICOM mode: Preamble length - (only these values are allowed) 00000 send 0 IDLE 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 22-9. SPI Transfer Format with SPMODE[CP] = 0

Figure 22-10 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



Note: Q = Undefined signal.

Figure 22-10. SPI Transfer Format with SPMODE[CP] = 1

22.3.1.1 SPI Examples with Different SPMODE[REV,LEN] Values

The examples below show how SPMODE[REV,LEN] effect 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 22-1.

```
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 22-2.

```
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 22-3.

```
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 22-4.

```
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 22-5.

```
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
```

22.3.2 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 22-11 shows both registers.

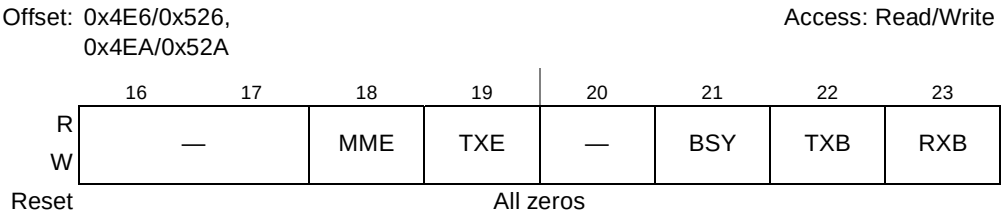


Figure 22-11. SPIE/SPIM—SPI Event/Mask Registers

Table 22-5 describes the SPIE/SPIM fields.

Table 22-5. SPIE/SPIM Field Descriptions

Bits	Name	Description
16–17	—	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.

22.3.3 SPI Command Register (SPCOM)

SPCOM, shown in Figure 22-12, is used to start SPI operation.

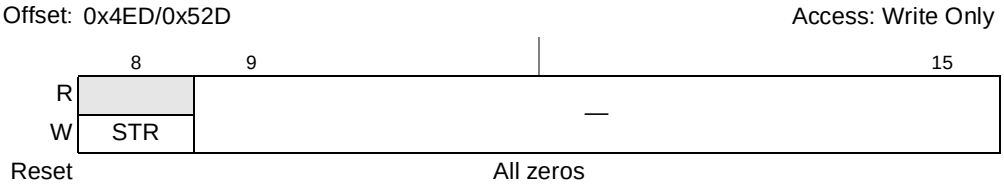


Figure 22-12. SPCOM—SPI Command Register

Table 22-6 describes the SPCOM fields.

Table 22-6. SPCOM Field Descriptions

Bits	Name	Description
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.
9–15	—	Reserved and should be cleared

22.4 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 22-7 shows the memory map of the SPI parameter RAM.

Table 22-7. 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 multi-user 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 22.4.1, "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.
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 Multi-user RAM.
2 **The user must initialize only items in bold.**

22.4.1 Receive/Transmit Bus Mode Registers

Figure 22-13 shows the fields in the receive/transmit bus mode registers. Bolded bits must be configured by the user.

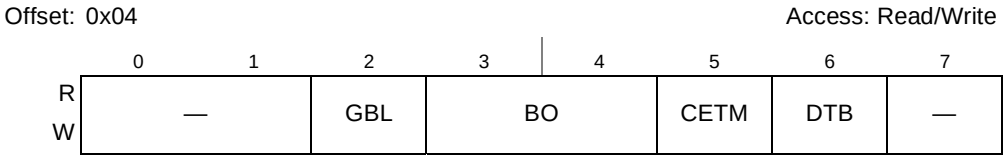


Figure 22-13. Rx/Tx Bus Mode Registers

Table 22-8 describes the Rx/Tx bus mode register fields.

Table 22-8. 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 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 19.1.8.2, “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.
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.

22.5 SPI Buffer Descriptor (BD) Table

As shown in Figure 22-14, 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 MPC8349 uses BDs to confirm reception and transmission or to indicate error conditions so that the CPU knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the multi-user RAM.

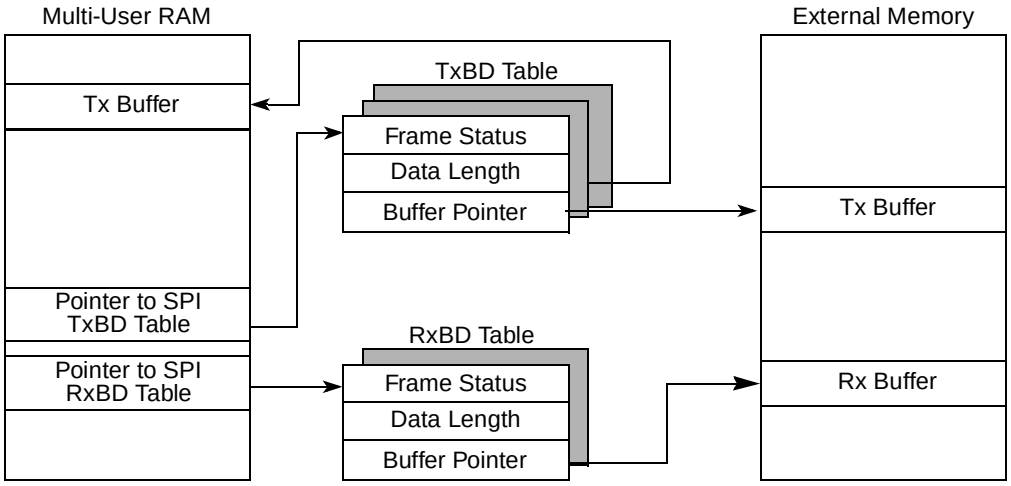


Figure 22-14. SPI Memory Structure

22.5.1 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 22-15 and Figure 22-16, has the following structure:

- The half word at offset + 0 contains status and control bits. The QUICC Engine module updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - For an RxBD, this is the number of octets the 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, since 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.
- The word at offset + 4 points to the beginning of the buffer.
  - For an RxBD, the pointer must be even and can point to internal or external memory.
  - For a TxBD, the pointer can be even or odd, unless the character exceeds 8 bits, for which it must be even. The buffer can be in internal or external memory.

22.5.1.1 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 CPU should write RxBD bits before the SPI is enabled. The format of an RxBD is shown in Figure 22-15.

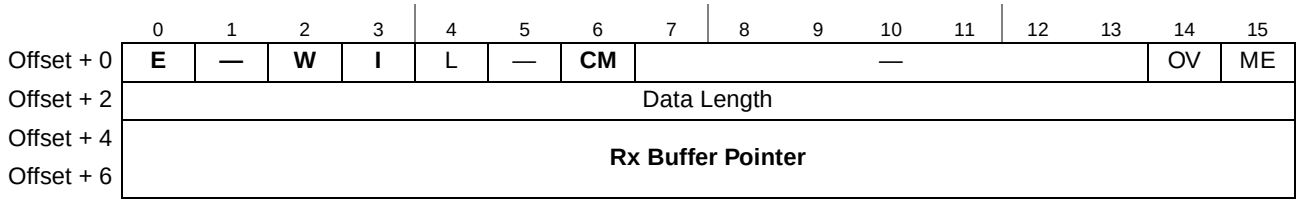


Figure 22-15. SPI RxBD

Table 22-9 describes the RxBD status and control fields.

Table 22-9. 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 CPU 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 CPU 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 multi-user 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 CPU 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 CPU 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
14	OV	Overrun. Set when a receiver overrun occurs during reception. Old received data is kept while the new data received is discarded.
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.

22.5.1.2 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 22-16.

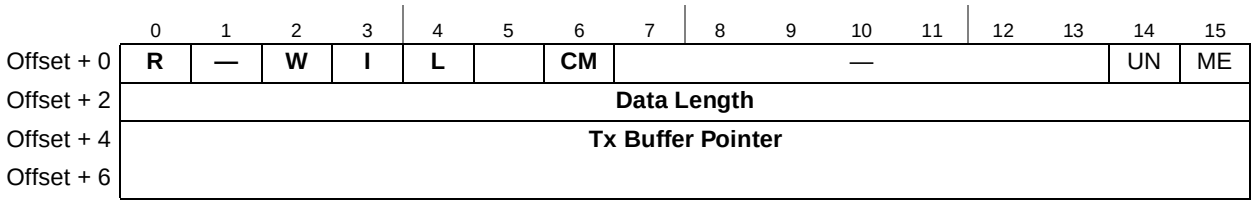


Figure 22-16. SPI TxBD

Table 22-10 describes the TxBD status and control fields.

Table 22-10. 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.
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

Table 22-10. SPI TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
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 No transmitter underrun condition encountered 1 Transmitter underrun condition encountered
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

22.6 SPI Commands

Table 22-11 lists transmit/receive commands sent to the QUICC Engine command register (CECR).

Table 22-11. 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.

22.7 SPI Programming Examples

22.7.1 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.

22.7.2 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 multi-user 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.

22.7.3 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, since 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.

22.8 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.

22.9 SPI in CPU Mode

Table 22-12. SPI Registers Summary in CPU Mode

Offset from CE_base	Register	Access	Reset Value
0x4E0/0x520	SPMODE–SPI mode register	R/W	0x0000_0000
0x4E6/0x526	SPIE–SPI event register	R/W	0x00
0x4EA/0x52A	SPIM–SPI mask register	R/W	0x00
0x4ED/0x52D	SPCOM–SPI command register	W	0x00
0x4F0/0x530	SPITD–SPI transmit register	W	0x0000_0000
0x4F4/0x534	SPIRD–SPI receive register	R	0xFFFF_FFFF

22.9.1 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.

22.9.2 SPI Mode Register (SPMODE) in CPU Mode

SPMODE, shown in Figure 22-17, controls both the SPI operation mode and clock source.

Offset: 0x4E0/0x520 Access: Read/Write

	0	1	2	3	4	5	6	7	8–11	12–15
R W	—	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN	PM
Reset	All zeros									

	16	17	18–19	20–24	25–31
R W	—	OP	—	CG	—
Reset	All zeros				

Figure 22-17. SPMODE-SPI Mode Register in CPU Mode

Table 22-13 describes the SPMODE fields.

Table 22-13. 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 22-9 and Figure 22-10. 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 22-9 and Figure 22-10. 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.
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.

Table 22-13. SPMODE Field Descriptions (continued)

Bits	Name	Description
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**
18–19	—	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

22.9.3 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 22-18 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 22-18. SPI Event Register (SPIE) in CPU Mode

Table 22-14 describes the SPIE fields in CPU mode.

Table 22-14. 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.

22.9.4 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 22-19 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 22-19. SPI Mask Register (SPIM) in CPU Mode

Table 22-15 describes the SPIM fields.

Table 22-15. 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

22.9.5 SPI Command Register (SPCOM) in CPU Mode

SPCOM, shown in Figure 22-20, indicates whether the next character to be written to SPITD is the last character of the frame.

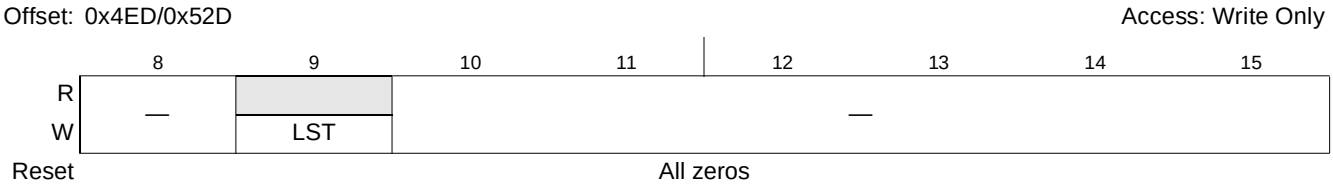


Figure 22-20. SPI Command Register (SPCOM) in CPU Mode

Table 22-16 describes the SPCOM fields.

Table 22-16. SPCOM Field Descriptions

Bits	Name	Description
8	—	Reserved. Should be cleared.

Table 22-16. SPCOM Field Descriptions

Bits	Name	Description
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.

22.9.6 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 22-21 shows the SPI transmit data register.

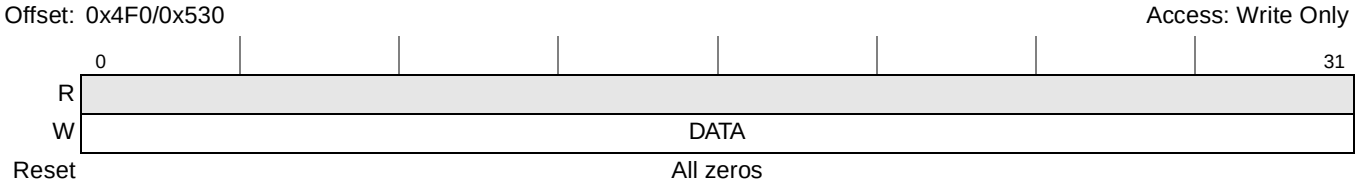


Figure 22-21. SPI Transmit Data Register (SPITD)

22.9.7 SPI Receive Data Register (SPIRD)

SPIRD holds a received character. Each time the SPIE[NE] bit is set, the CPU can read the SPIRD.

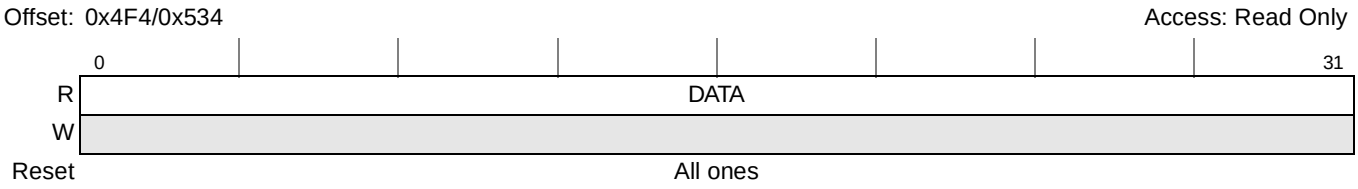


Figure 22-22. SPI Receive Data Register (SPIRD)

22.9.7.1 SPIRD Examples with Some SPMODE[REV,LEN] Values

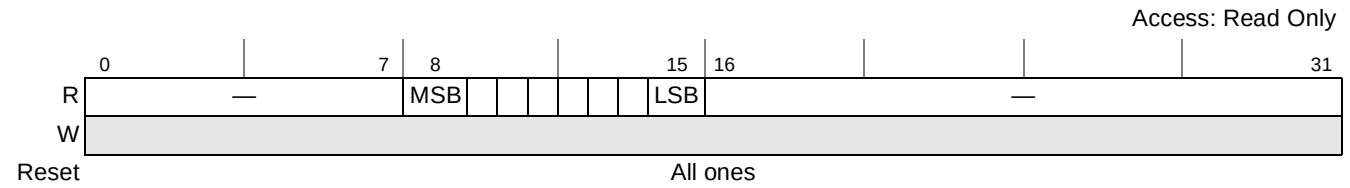


Figure 22-23. SPIRD in REV=1 LEN=0x7

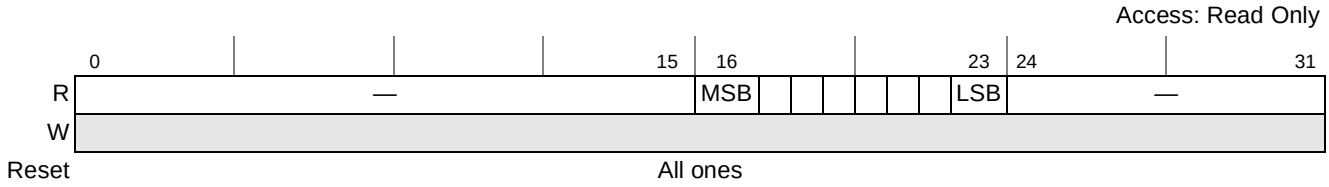


Figure 22-24. SPIRD in REV=0 LEN=0x7

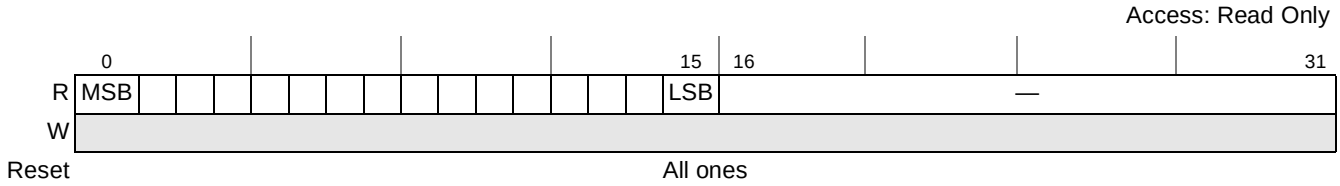


Figure 22-25. SPIRD in REV=1 LEN=0xF

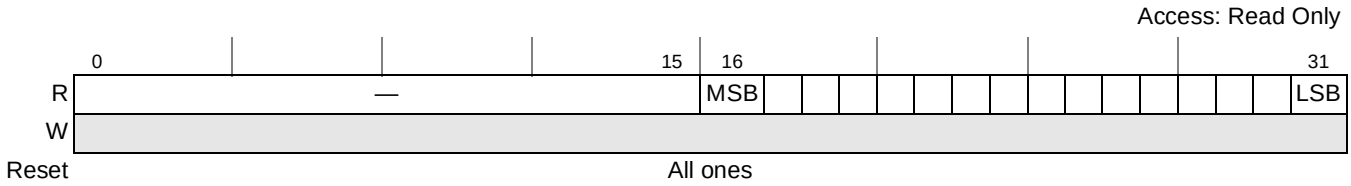


Figure 22-26. SPIRD in REV=0 LEN=0xF

Chapter 23
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, QMC, and EFM. The supported interfaces include RS-232, MII/RMII, GMII/RGMII, UTOPIA L2, and POS-PHY L2. UCC1 and UCC2 also support TBI/RTBI interfaces to SerDes. Each UCC can also be connected to the TSA in HDLC, Transparent, QMC, Serial ATM, or EFM protocols.

23.1 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 Quick 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 1 Gbps (Gigabit Ethernet) full duplex.

The protocols in the UCC are split into two categories:

- Slow Protocols:
  - UART
  - BISYNC
  - QMC
  - Serial ATM
- Fast Protocols:
  - HDLC
  - Transparent
  - 10/100/1000 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; this difference allows the UCC to run at OC-12 or 1 Gbps rates. Figure 23-1 shows the UCC block diagram.

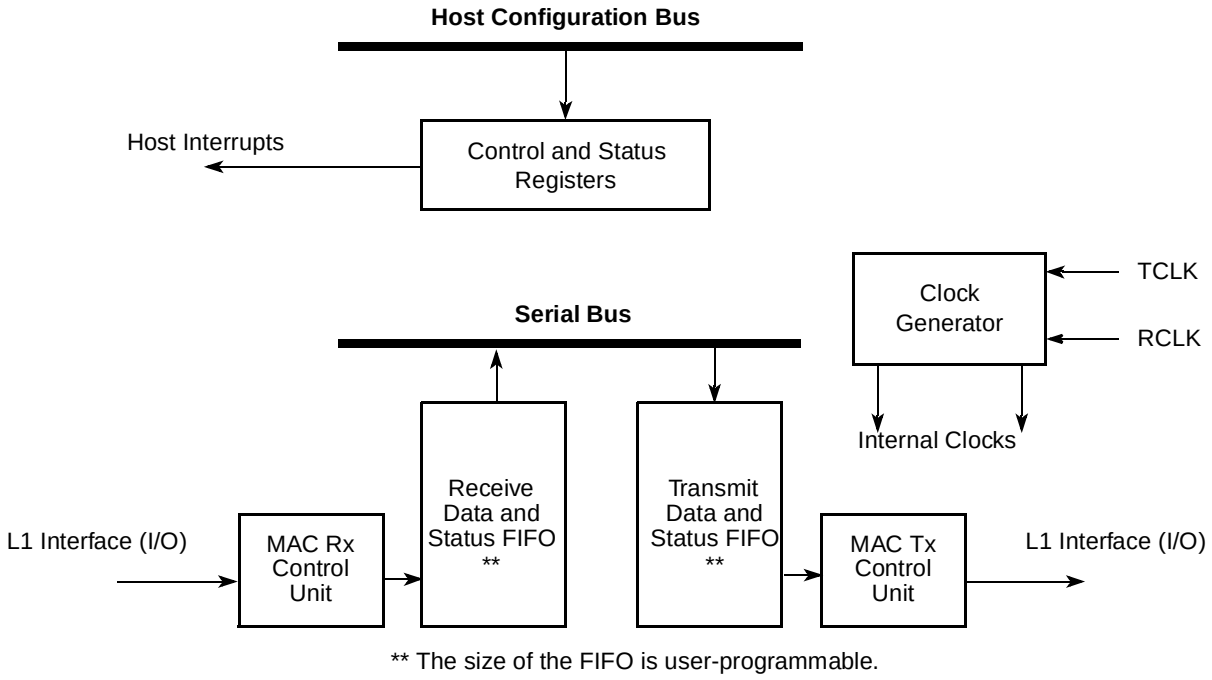


Figure 23-1. UCC Block Diagram

23.2 UCC Feature Set

The UCC feature set includes:

- HDLC/SDLC, HDLC bus, and transparent in single and nibble bit modes
- 10/100/1000 802.3 Ethernet through MII, RMII, GMII, RGMII interfaces
- Gigabit Ethernet through TBI and RTBI interfaces for UCC1 and UCC2.
- ATM up to OC-12 (622 Mbps) rate AAL0,1,5 through UTOPIA L2 8/16 bit interface
- ATM up to OC-3 (155 Mbps) rate AAL2 through UTOPIA L2 8-/16-bit interface
- POS up to STM-4 (622 Mbps) rate through a POS-PHY L2 16-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, POS protocols

- 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.

23.2.1 UCC Base Addresses

23.2.1.1 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 23-1.

The UCC Parameter RAM base addresses are programmable using the ASSIGN PAGE host command. See Section 20.3.1.1.1, "Assign Page Command," for details.

NOTES: Backward Compatibility

UCC1-UCC3 are backward-compatible to FCC1-FCC3, and UCC5-UCC8 are backward-compatible to SCC1-SCC4 in both their parameter RAM base address, and in their respective protocols (UCC1-UCC4 are set for fast protocols, and UCC5–UCC8 are 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 23-1), but the user may relocate the pages to allow for improved optimization of the memory 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 23-1. 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
6	0x8100	UCC6	256
7	0x8200	UCC7	256
8	0x8300	UCC8	256

1 Offset from RAM_Base.

23.2.1.2 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 23-2.

Table 23-2. UCC Register Base Addresses

Peripheral	Address[1]	Size (Bytes)
UCC1	0x2000	512
UCC2	0x3000	512
UCC3	0x2200	512
UCC4	0x3200	512
UCC5	0x2400	512
UCC6	0x3400	512
UCC7	0x2600	512
UCC8	0x3600	512

1 Offset from QUICC Engine module Base

23.3 Programming Model

23.3.1 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 24, "UCC as Slow Communications Controllers." For fast protocols, see Chapter 27, "UCC for Fast Protocols."

Table 23-3. 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
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

1 Offset from UCCx base.

23.3.2 General UCC Extended Mode Register (GUEMR)

GUEMR, shown in Figure 23-2, configures the UCC to operate as a fast or a slow communication controller. Also, it has ATM protocol-specific mode settings.

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.

Address: UCC*x*_Base + 0x90 Access: Read/Write

	0			3		5	6	7
R	—						URMODE	UTMODE
W								
Reset	0	0	0	1	0	0	*n*[1]	*n*[1]

Figure 23-2. General UCC Extended Mode Register

1 See Table 23-5.

Table 23-4 describes the GUEMR fields.

Table 23-4. 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:** In most applications, 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 23-5. 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
UCC6	0x10	Slow protocols
UCC7	0x10	Slow protocols
UCC8	0x10	Slow protocols

23.3.3 UCC Transmit Polling Timer (UTPT)

The UTPT, shown in Figure 23-3, 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

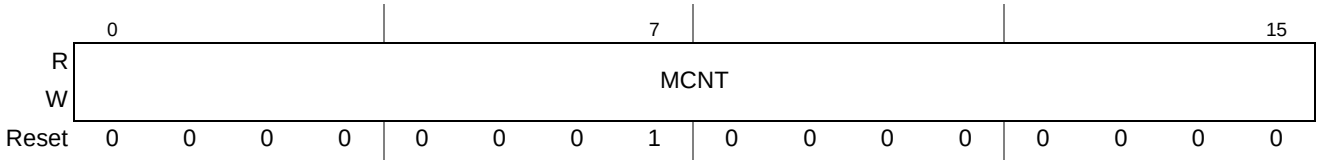


Figure 23-3. UCC Transmit Polling Timer

The UTPT register is described in Table 23-6.

Table 23-6. 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.

23.3.4 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 23-4.

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.

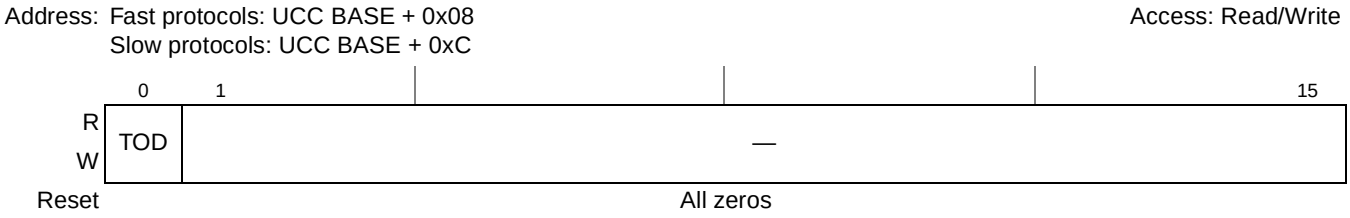


Figure 23-4. UCC Transmit-on-Demand Register

Fields in the UTODR are described in Table 23-7.

Table 23-7. 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.

23.3.5 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.

23.3.6 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 19.3.11, “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 (i.e., 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 19.3.10, “QUICC Engine System Interrupt Pending Register (CIPNR).”

23.3.7 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).

23.4 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 23-8). 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 23-8. 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 19.2, “Interrupt Controller.”

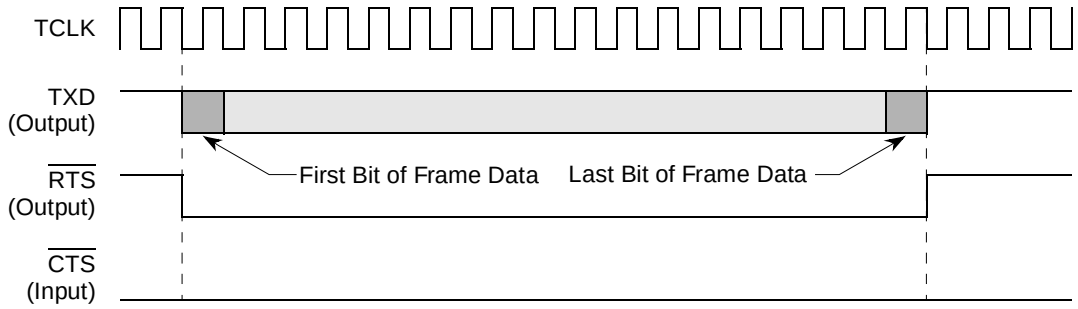
23.5 Controlling UCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

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.

23.5.1 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.

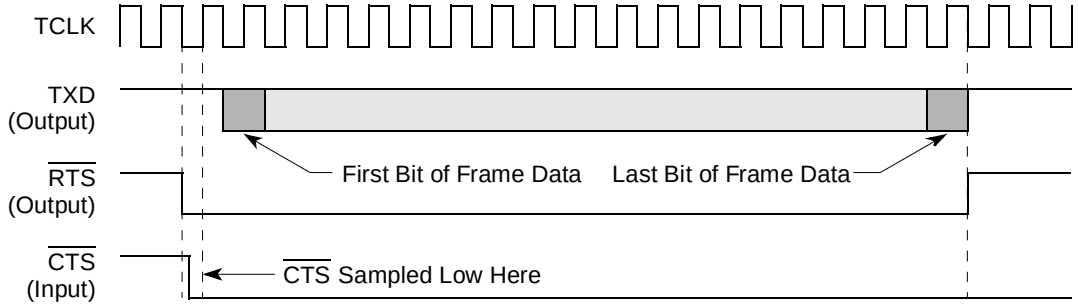
Figure 23-5 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit cycles, regardless of the status of CTSS. This operation assumes that $\overline{\text{CTS}}$ is already asserted to the UCC or that $\overline{\text{CTS}}$ is reprogrammed to be a general purpose parallel I/O line, in 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.



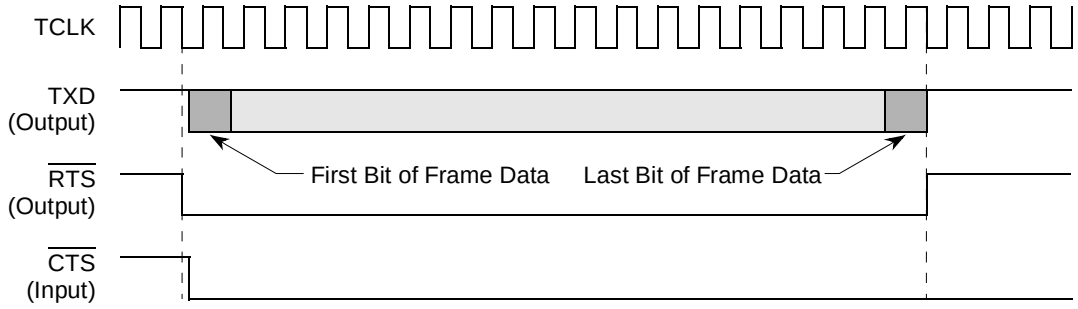
NOTE:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 23-5. Output Delay from RTS Asserted for Synchronous Protocols

When RTS is asserted, if CTS is not already asserted, delays to the first data bit depend on when CTS is asserted. Figure 23-6 shows that the delay between CTS and the data can be approximately 0.5 to 1 bit cycles or no delay, depending on the status of CTSS.



NOTE:
1. CTSS = 0. CTSP is a don't care.



NOTE:
1. CTSS = 1. CTSP is a don't care.

Figure 23-6. Output Delay from CTS Asserted for Synchronous Protocols

If CTS is programmed to envelope data, negating it during frame transmission causes a CTS lost error. Negating CTS forces RTS high and Tx data to become idle. If CTSS is zero, the UCC must sample CTS before a CTS lost is recognized; otherwise, the negation of CTS immediately causes the CTS lost condition. See Figure 23-7.

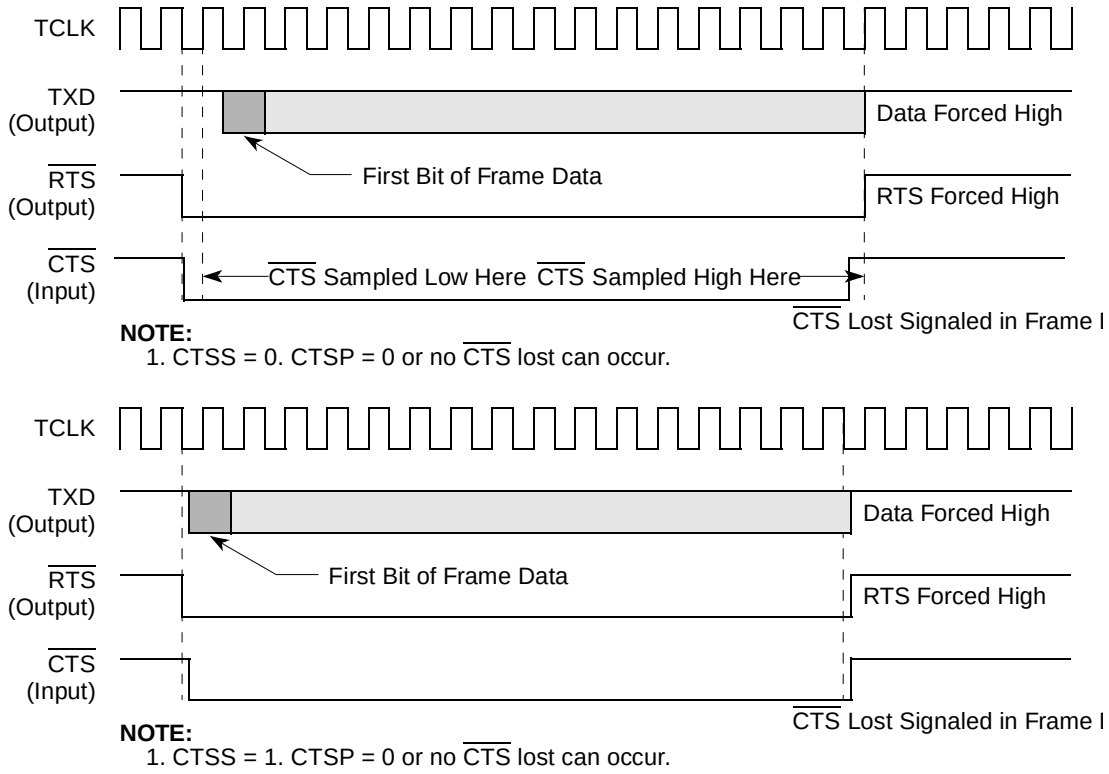


Figure 23-7. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

Note that if CTSS = 1, $\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as shown in Figure 23-8. If CDS is zero, $\overline{\text{CD}}$ is sampled on the rising Rx clock edge before data is received. If CDS is 1, $\overline{\text{CD}}$ transitions cause data to be immediately gated into the receiver.

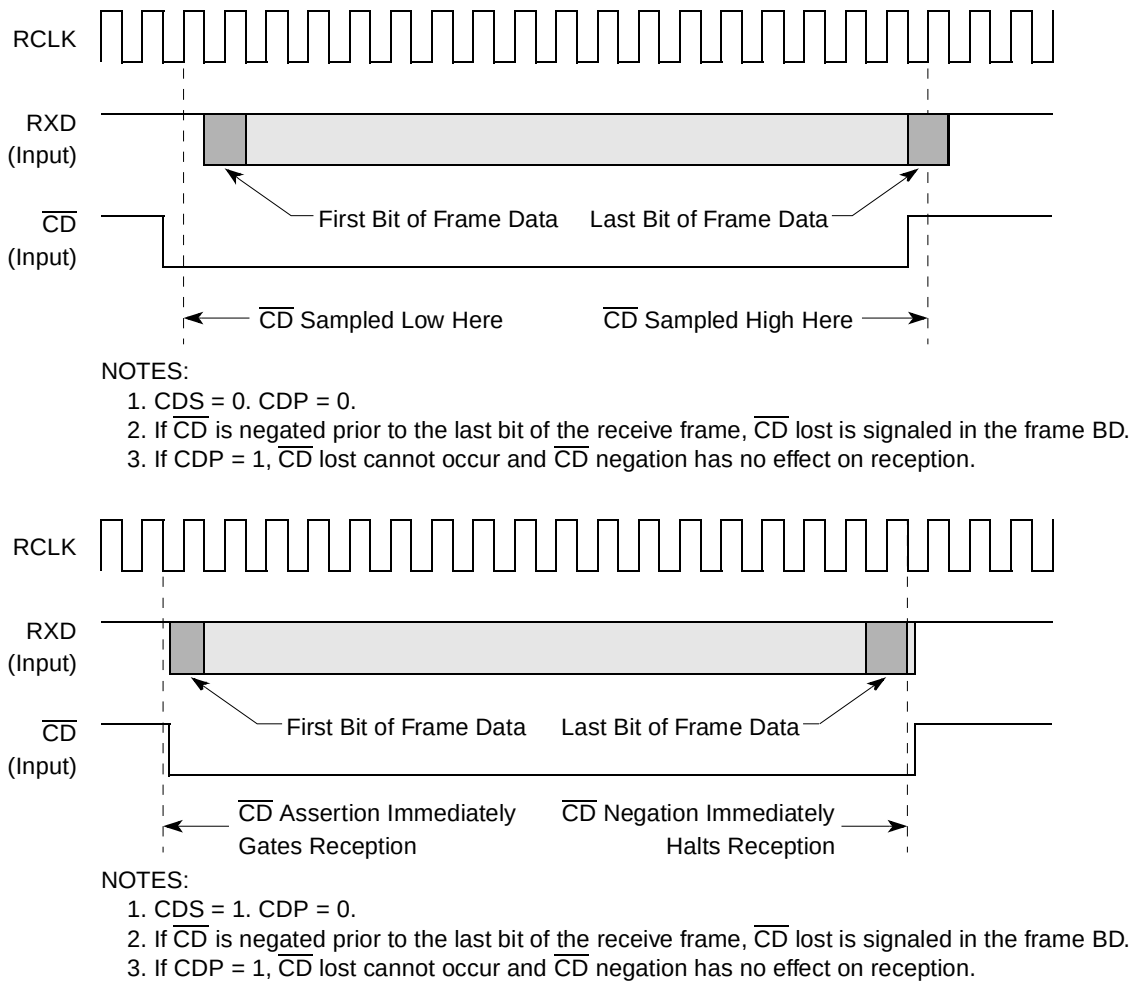


Figure 23-8. Using $\overline{\text{CD}}$ to Control Synchronous Protocol Reception

If $\overline{\text{CD}}$ is programmed to envelope the data, it must remain asserted during frame transmission or a $\overline{\text{CD}}$ lost error occurs. Negation of $\overline{\text{CD}}$ terminates reception. If CDS is cleared, the UCC must sample $\overline{\text{CD}}$ before a $\overline{\text{CD}}$ lost error is recognized; otherwise, the negation of $\overline{\text{CD}}$ immediately causes the $\overline{\text{CD}}$ lost condition. If CDS is set, all $\overline{\text{CD}}$ transitions must occur while the Rx clock is low.

23.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{\text{RTS}}$ is asserted when UCC data is loaded into the Tx FIFO and a falling Tx clock occurs. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ flow control as described in Chapter 25, "UCC 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.

23.5.3 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 23-9 shows the different encoding methods.

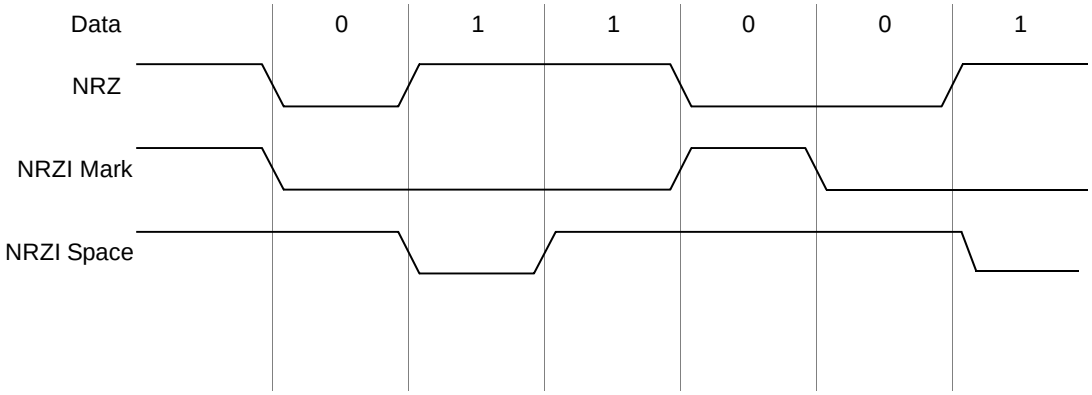


Figure 23-9. Data Encoding Examples

NRZ or NRZI codings can be selected in RENC/TENC and RINV/TINV. Coding definitions are shown in Table 23-9.

Table 23-9. 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.

[1] Available only for slow protocols.

23.6 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 27.4.2, “UCC Registers in Fast Mode”).
- Slow protocols: GUMR_L[MODE], GUMR_H[TTX], GUMR_H[TRX] (Section 24.2.2, “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 23.3.2, “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 23-10. 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
POS	1	1	Fast	1110	0	0
HDLC	1	1	Fast	0000	0	0
Transparent	1	1	Fast	0000	1	1
EFM	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

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.

23.7 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 or POS), 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 24
UCC as Slow Communications Controllers

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 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 25, "UCC UART Mode and Asynchronous HDLC"
- Chapter 26, "UCC BISYNC Mode"
- Chapter 42, "QMC (QUICC Multi-Channel Controller)"
- Chapter 38, "Serial ATM Microcode"

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.

24.1 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

24.2 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.

24.2.1 UCC Memory Map for Slow Protocols

Table 24-1. 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.

24.2.2 General UCC Mode Registers (GUMR)

Each UCC contains a GUMR that defines options common to most of the protocols. GUMR_L, shown in Figure 24-1, contains the low-order 32 bits; GUMR_H, shown in Figure 24-2, contains the high-order 32 bits. Some GUMR operations are described in later sections.

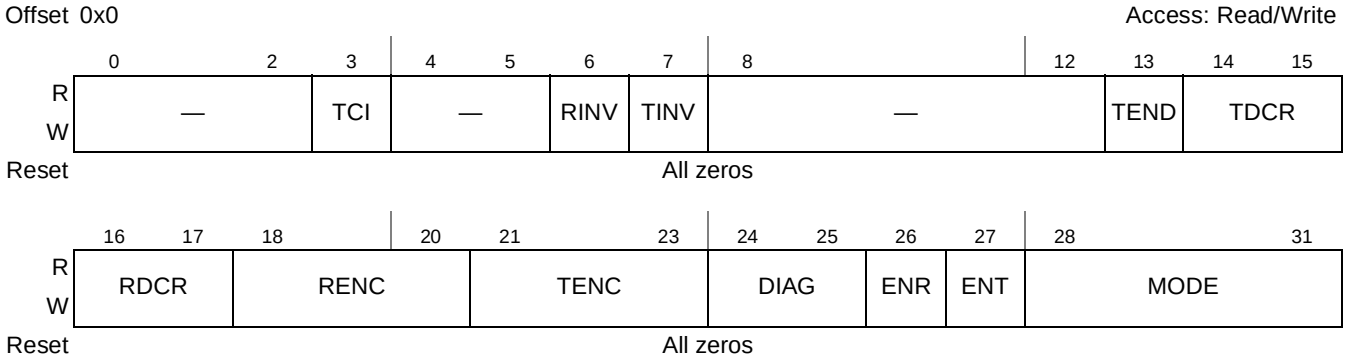


Figure 24-1. GUMR_L—General UCC Mode Register Low Order

Table 24-2. 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. **Note:**
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.
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	

Table 24-2. GUMR_L Field Descriptions (continued)

Bit	Name	Description
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 23.5, "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 36.6.4, "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 24.4.6, "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 24.4.6, "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 24-2. 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 25, “UCC UART Mode and Asynchronous HDLC”) 1000 BISYNC All other values are reserved

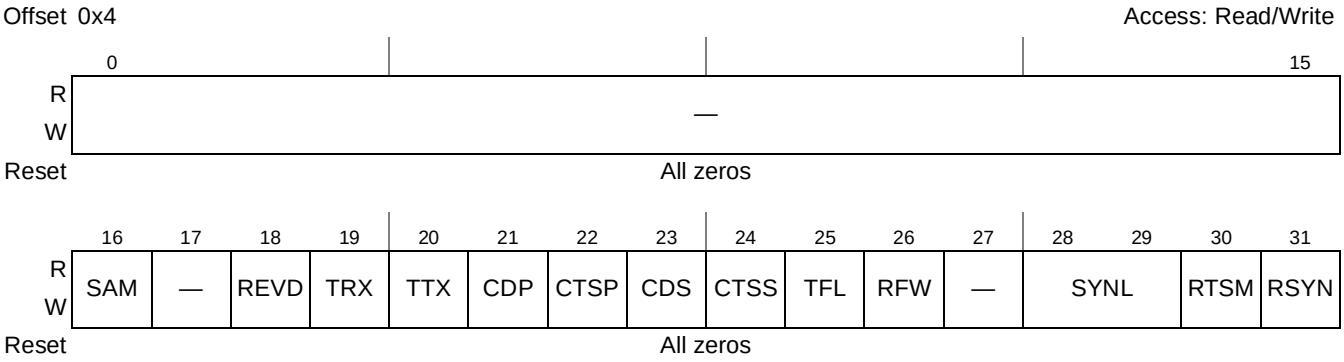


Figure 24-2. GUMR_H—General UCC Mode Register (High Order)

Table 24-3 describes GUMR_H fields.

Table 24-3. 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 26.3.2.5, “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 24-3. 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 since 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 24.2.3, "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 24-3. 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.

24.2.3 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 24-3 shows the sync fields.

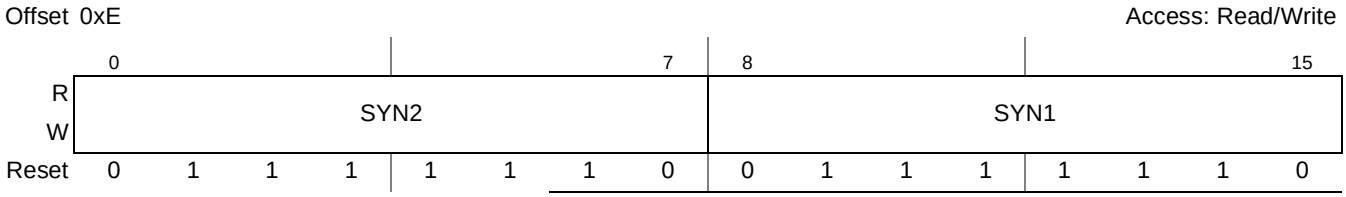


Figure 24-3. Data Synchronization Register (UDSR)

24.2.4 UCC Transmit Polling Timer (UTPT)

Refer to Section 23.3.3, “UCC Transmit Polling Timer (UTPT).”

24.2.5 UCC Transmit-on-Demand Register (UTODR)

Refer to Section 23.3.4, “UCC Transmit-On-Demand Register (UTODR).”

24.3 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 24-4, is the same in each UCC mode. Only the status and control bits differ for each protocol.

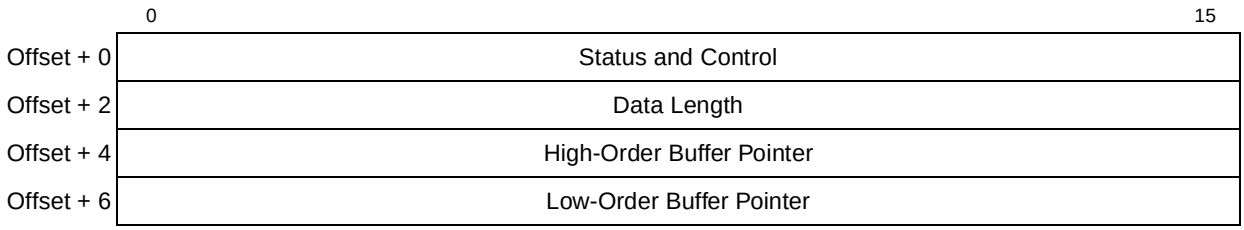


Figure 24-4. UCC Buffer Descriptors (BDs)

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 24-5 shows the UCC BD table and buffer structure.

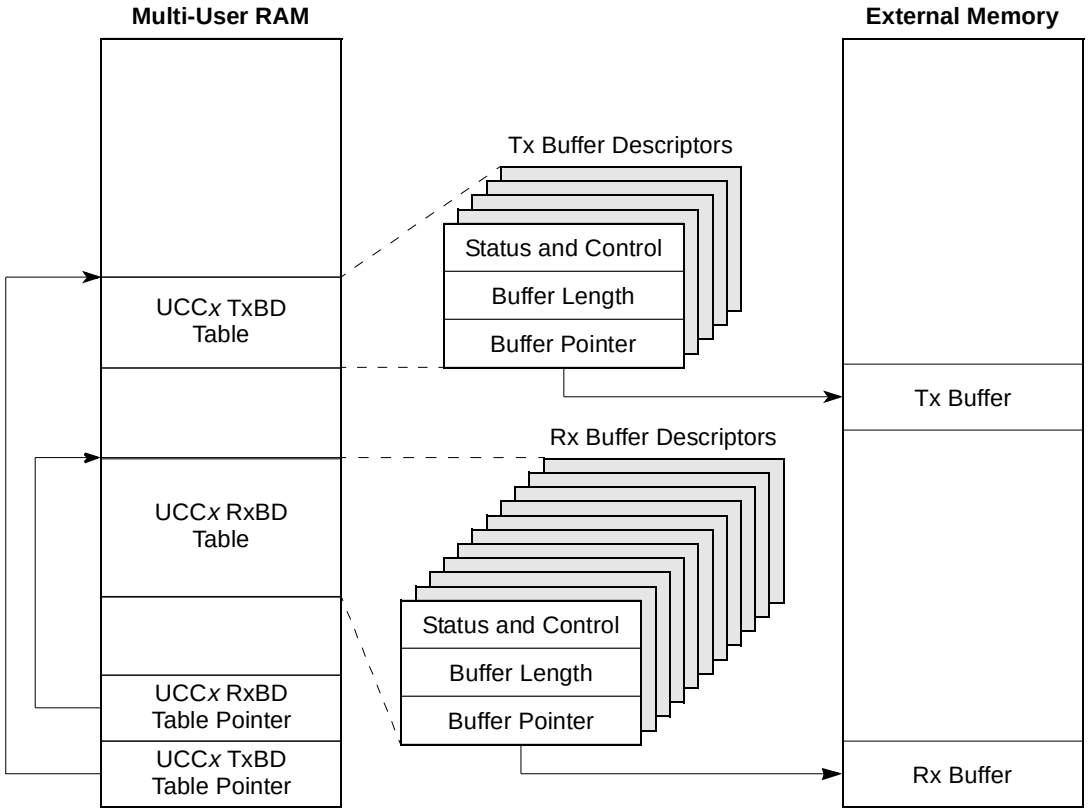


Figure 24-5. UCC BD and Buffer Memory Structure

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.

24.4 UCC Parameter RAM

The UCC parameter RAM base address is defined in Section 23.2.1.1, "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 24-4.

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 24.4.6, "Reconfiguring the UCC."

Table 24-4 shows the parameter RAM map for all UCC protocols. Boldfaced entries must be initialized by the user.

Table 24-4. 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 24.4.1, "Bus Mode Registers (RBMR and TBMR)."
0x05	**TBMR**	Byte	Tx bus mode. See Section 24.4.1, "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]

Table 24-4. UCC Parameter RAM Map for All Protocols (continued)

Offset [1]	Name	Width	Description
0x0C		Word	Rx internal buffer pointer[2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	Current RxBD pointer. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the BD table is reached, the 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.)

[1] From UCC base.

[2] These parameters need not be accessed for normal operation but may be helpful for debugging.

[3] For QUICC Engine module use only.

24.4.1 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 24-6 shows the register format. Boldfaced entries must be initialized by the user.

Address: UCC*x* base + 0x04 (RBMR*x*); Access: Read/Write
UCC*x* base + 0x05 (TBMR*x*)

	0	1	2	3	4	5	6	7
R W	—		**GBL**	**BO**		**CETM**	**DTB**	**BDB**
Reset	All zeros							

Figure 24-6. Bus Mode Registers (RBMR and TBMR)

Table 24-5 describes RBMR*x*/TBMR*x* fields.

Table 24-5. 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 19.1.8.2, "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.
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.

24.4.2 UCC Event Register (UCCE)

Refer to Section 23.3.5, "UCC Event Register (UCCE)," and to the protocol specific chapters of the user manual.

24.4.3 UCC Mask Register (UCCM)

Refer to Section 23.3.6, "UCC Mask Register (UCCM)," and to the protocol specific chapters of the user manual.

24.4.4 UCC Status Register

.Refer to Section 23.3.7, “UCC Status Register,” and to the protocol specific chapters of the user manual.

24.4.5 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].

24.4.6 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.

24.4.6.1 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. If the UCC receiver is enabled, a GRACEFUL STOP RECEIVE command should be issued.
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 and a RESTART RECEIVE command.
5. Set GUMR_L[ENT]. Transmission begins using the TxBD pointed to by TBPTR, assuming the R bit is set.

24.4.6.2 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].

24.4.6.3 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 24.4.6.5, “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.

24.4.6.4 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].

24.4.6.5 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.

24.4.7 Saving Power

To save power when not in use, a UCC can be disabled by clearing GUMR_L[ENT, ENR].

Chapter 25
UCC UART Mode and Asynchronous HDLC

25.1 Introduction

The universal asynchronous receiver transmitter (UART) protocol is commonly used to send low-speed data between devices. The term asynchronous is used because it is not necessary to send clocking information along with the data being sent. UART links are typically 38,400 baud or less and are character-based. Asynchronous links are used to connect terminals with other devices. Even where synchronous communications are required, the UART is often used as a local port to run board debugger software. The character format of the UART protocol is shown in Figure 25-1.

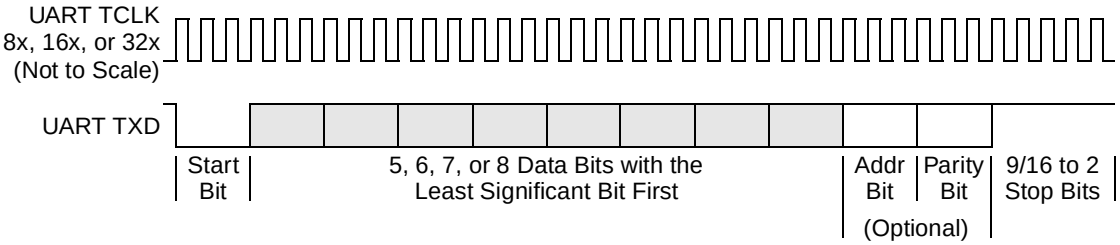


Figure 25-1. UART Character Format

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.

25.1.1 Block Diagram

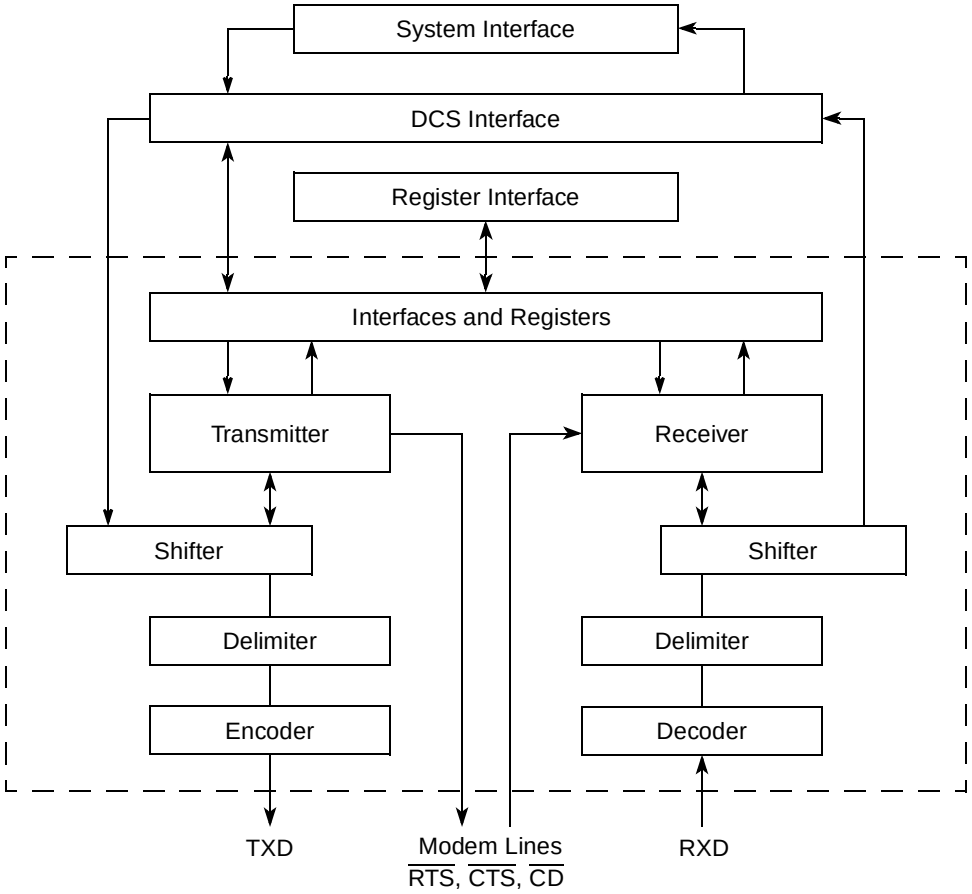


Figure 25-2. UART Block Diagram

25.1.2 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

25.1.3 Modes of Operation

- Asynchronous mode
- Synchronous mode

25.1.3.1 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.

25.1.3.2 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.

25.2 External Signal Descriptions

Table 25-1. 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

Table 25-1. Interface A—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{CTS}}$	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

25.3 Memory Map/Register Definition

25.3.1 Overview

Table 25-2. Memory Map

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x30[1]	UCC UART parameter RAM	R/W	0x0000_0000	25.3.2/25-5
As set in parameter ram	UCC UART receive buffer descriptor (RxBD)	R/W	0x0000_0000	25.3.5/25-10
As set in parameter ram	UCC UART transmit buffer descriptor (TxBD)	R/W	0x0000_0000	25.3.6/25-13
0xE[2]	UDSR data synchronization register (UDSR)	R/W	0x7E_7E	25.3.3/25-8
0x8[2]	UART mode register (UPSMR)	R/W	0x0000_0000	25.3.4/25-8
0x10[2]	UCC UART event register (UCCE)	R/W	0x0000_0000	25.3.7/25-14
0x14[2]	UCC UART mask register (UCCM)	R/W	0x0000_0000	25.3.7/25-14
0x17[2]	UCC UART status register (UCCS)	R/W	0x0000_0000	25.3.8/25-16

1 Offset from UCC's parameter RAM base address

2 Offset from UCC's register base address.

25.3.2 UCC UART Parameter RAM

For UART mode, the protocol-specific area of the UCC parameter RAM is mapped as shown in Table 25-3. Description of contents of the first 0x30 bytes can be found in Chapter 24, "UCC as Slow

Communications Controllers.” 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 20.2, “Parameter RAM.”

Note: Bit fields in **boldface** must have their initial values set by the user.

Table 25-3. 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.
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 25.5.4, “Inserting Control Characters into the Transmit Data Stream.”

Table 25-3. UART-Specific UCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
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.
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.

[1] From UCC_base. See Section 20.2, “Parameter RAM,” on page 20-1.

25.3.3 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 25-3. UDSR Register

Table 25-4. 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 25-3

25.3.4 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 25-4 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 25-4. Protocol-Specific Mode Register for UART (UPSMR)

Table 25-5 describes PSMR UART fields.

Table 25-5. UPSMR UART Field Descriptions

Bits	Name	Description
0	FLC	Flow control. The GUMR_H[CTSS,CTSP] (see Chapter 8) 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 25-5. 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.

25.3.5 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 25-5 shows an example of how RxBDs are used in receiving.

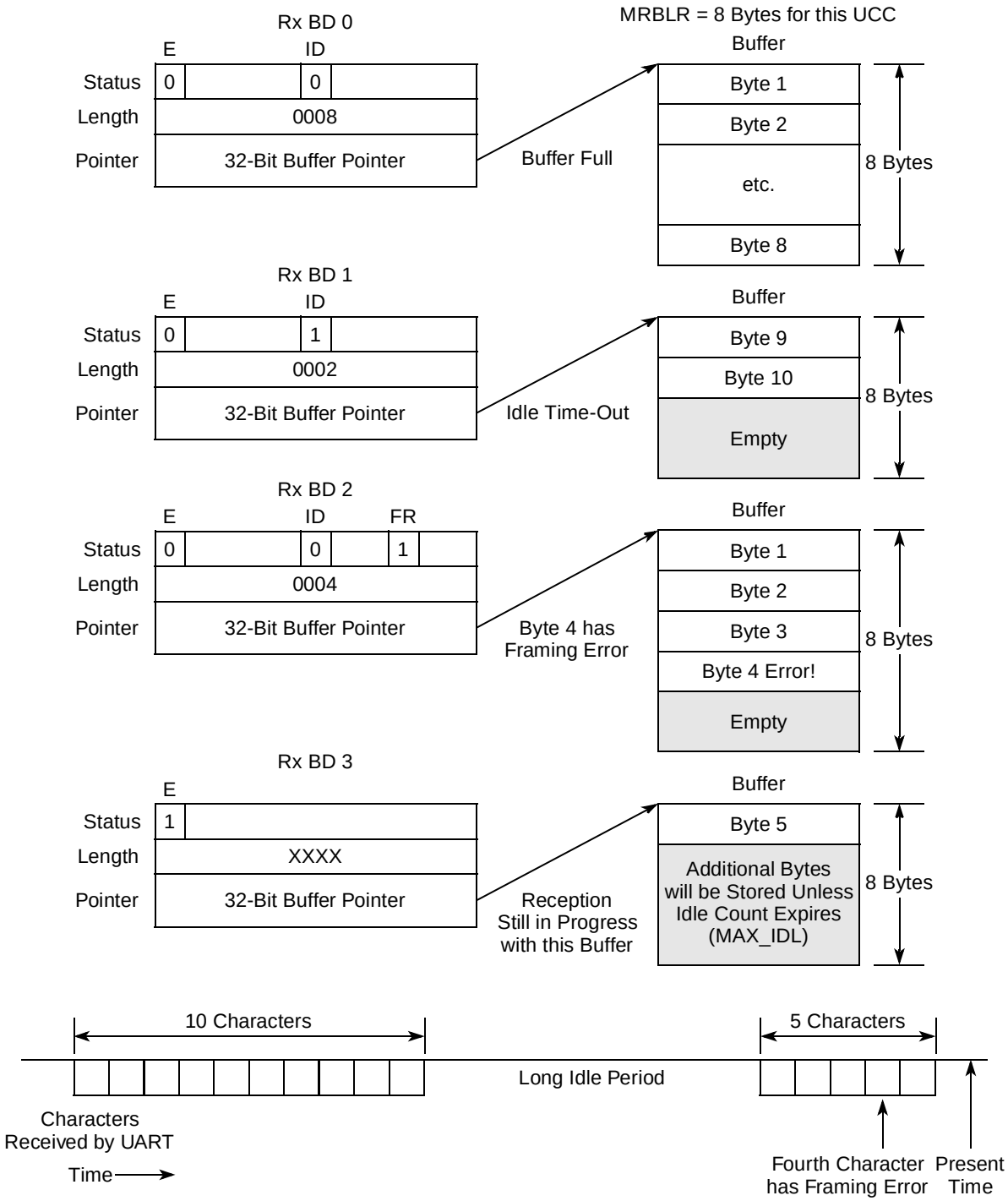


Figure 25-5. UCC UART Receiving using RxBDs

Figure 25-6 shows the UCC UART RxBD.

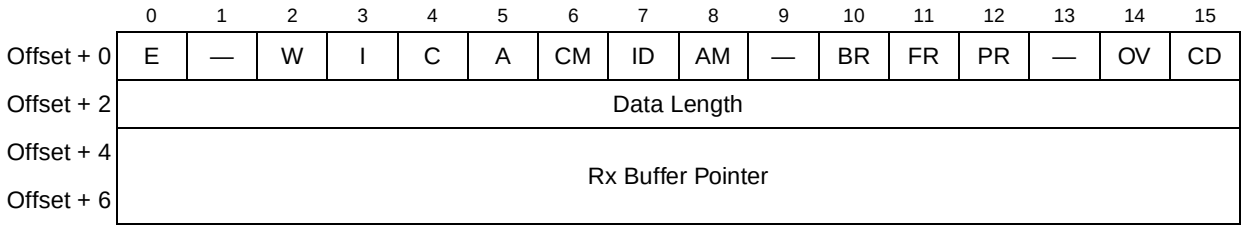


Figure 25-6. UCC UART Receive Buffer Descriptor (RxBD)

Table 25-6 describes RxBD status and control fields. Fields shown in bold type must have their initial values set by the user.

Table 25-6. 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 25-6. 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 24.3, "UCC Buffer Descriptors (BDs)," describes the data length and buffer pointer fields.

25.3.6 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 25-7 shows the UCC UART TxBD.

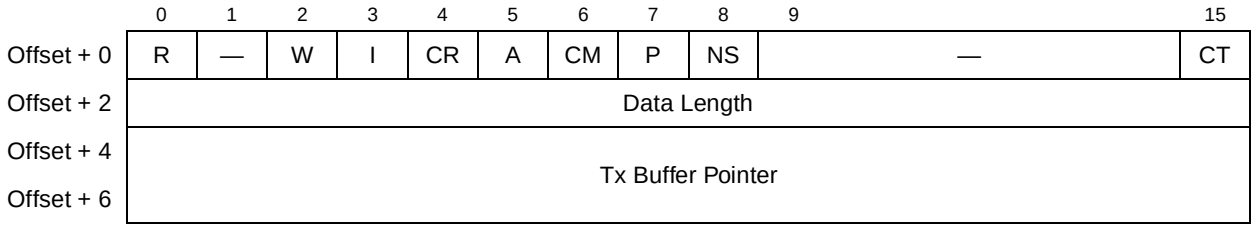


Figure 25-7. UCC UART Transmit Buffer Descriptor (TxBD)

Table 25-7 describes TxBD status and control fields. Fields shown in bold type must have their initial values set by the user.

Table 25-7. 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

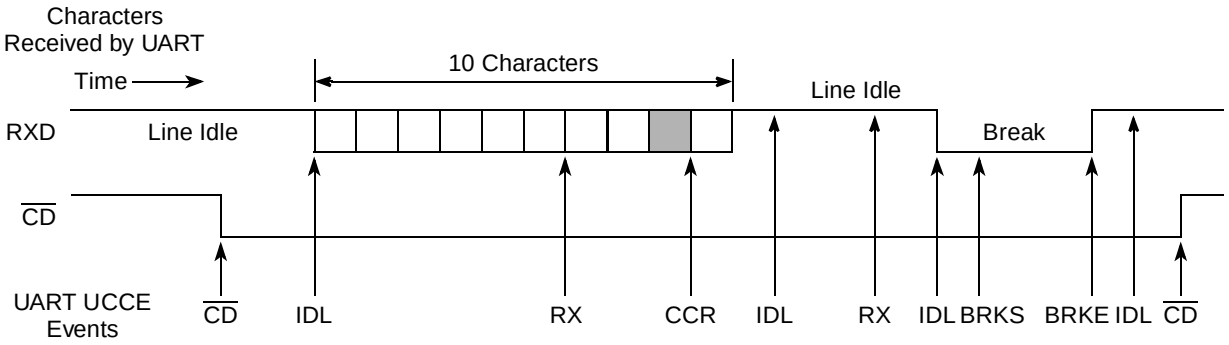
Table 25-7. UCC UART TxBD Status and Control Field Descriptions (continued)

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 25.5.7, “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 24.3, “UCC Buffer Descriptors (BDs).”

25.3.7 UCC UART Event Register (UCCE) and Mask Register (UCCM)

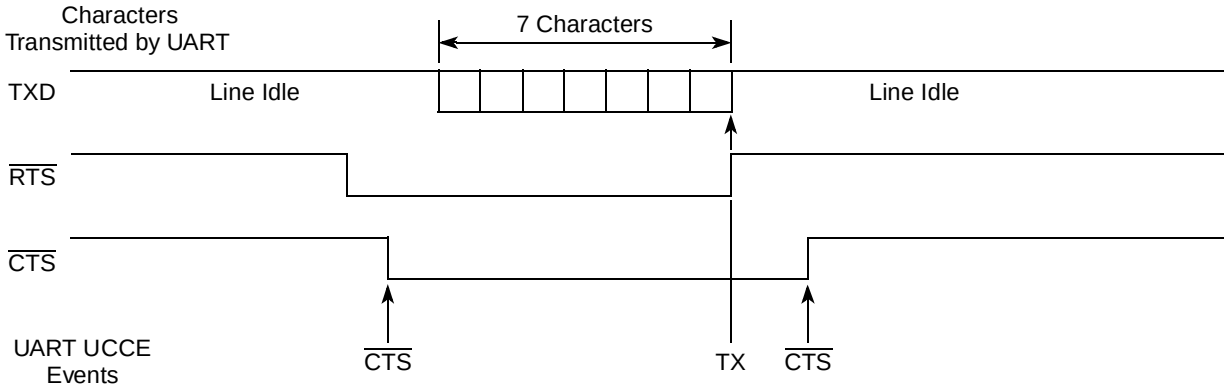
UCCE is used to report events recognized by the UART channel and to generate interrupts. When an event is recognized, the controller sets the corresponding UCCE bit. Interrupts can be masked in UCCM, which has the same format as UCCE. Setting a mask bit enables the corresponding UCCE interrupt; clearing a bit masks it. Figure 25-8 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 CD event must be programmed in the 'QUICC Engine ports interrupts' in the system IPIC, 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 CTS event must be programmed in the 'QUICC Engine ports interrupts' in the system IPIC, not in the UCC itself.

Figure 25-8. 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 25-9 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	—	AB	IDL	GRA	BRKE	BRKS	—	CCR	BSY	TX	RX
W											
Reset	All zeros										

Figure 25-9. UCC UART Event Register (UCCE) and Mask Register (UCCM)

Table 25-8 describes UCCE fields for UART mode.

Table 25-8. 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 21, "QUICC Engine 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.

1 Reserved bits in the UCCE should not be masked in the UCCM register.

25.3.8 UCC UART Status Register (UCCS)

UCCS, shown in Figure 25-10, monitors the real-time status of RXD.

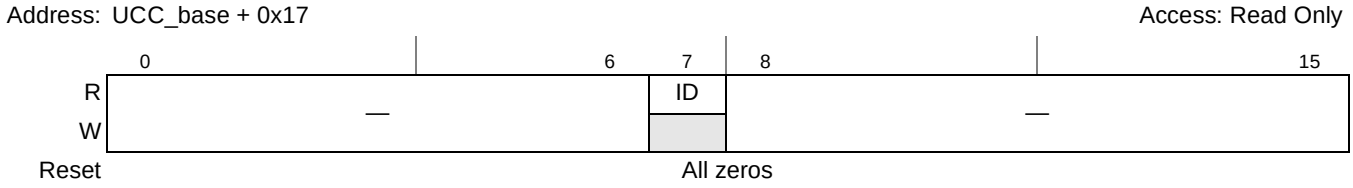


Figure 25-10. UCC Status Register for UART Mode (UCCS)

Table 25-9 describes UART UCCS fields.

Table 25-9. 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.

25.4 Functional Description

25.4.1 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 25.5.2, “Receiving Control Characters.”

25.4.2 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.

25.5 UCC UART Commands

The transmit commands in Table 25-10 are issued to the QUICC Engine command register (CECR).

Table 25-10. 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 25-11.

Table 25-11. 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.

25.5.1 Multidrop Systems and Address Recognition

In multidrop systems, more than two stations can be on a network, each with a specific address. Figure 25-11 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.

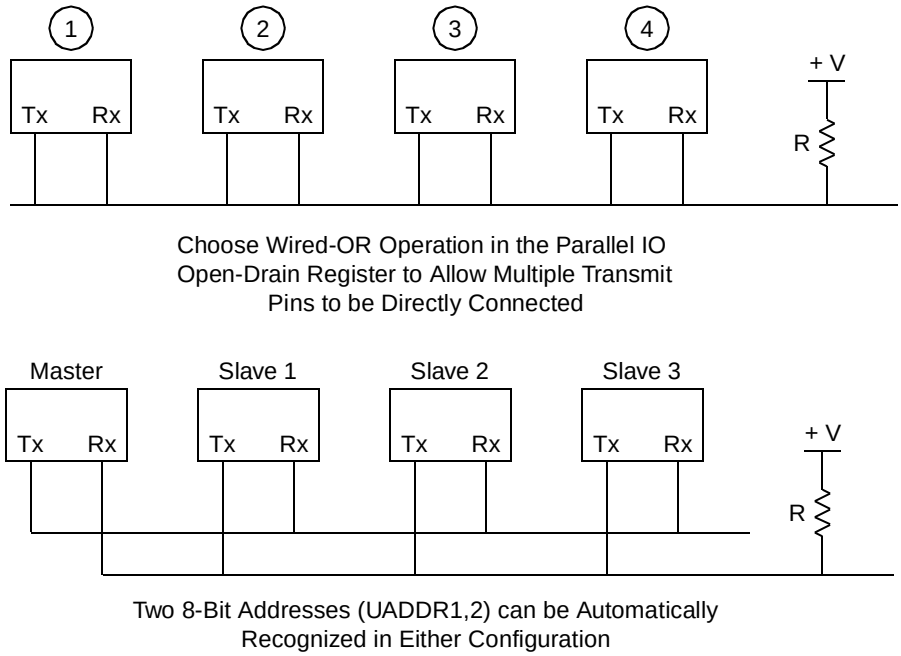


Figure 25-11. Two UART Multidrop Configurations

25.5.2 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 25-12.

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 25-12. Control Character Table

Table 25-12 describes the data structure used in control character recognition. Items in bold must be initialized by the user.

Table 25-12. 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.

25.5.3 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.

25.5.4 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 25-13, 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.

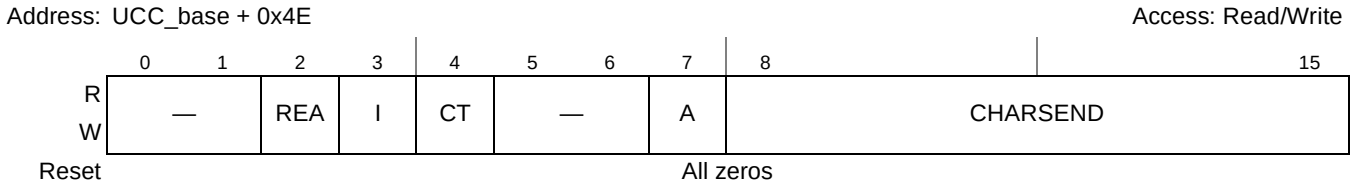


Figure 25-13. Transmit Out-of-Sequence Register (TOSEQ)

Table 25-13 describes TOSEQ fields. Fields shown in bold type should be initialized by the user.

Table 25-13. TOSEQ Field Descriptions

Bit	Name	Description
0–1	—	Reserved, should be cleared.
2	**REA**	Ready. Set when the character is ready for transmission. Remains 1 while the character is being sent. The 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 25-13. TOSEQ Field Descriptions (continued)

Bit	Name	Description
7	**A**	Address. Setting this bit indicates an address character for multidrop mode.
8–15	**CHARSEND**	Character send. Contains the character to be sent. Any 5- to 8-bit character value can be sent in accordance with the UART configuration. The character should be placed in the lsbs of CHARSEND. This value can be changed only while REA = 0.

25.5.5 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.

25.5.6 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.

25.5.7 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.

25.5.8 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 25-14.

Table 25-14. 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 25.3.4, "UART Mode Register (UPSMR)."

Reception errors are described in Table 25-15.

Table 25-15. 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.
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.

25.6 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

25.7 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 25.9, “Transmitter Transparency Encoding”

Figure 25-14. 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.

25.8 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 25.10, “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.

25.9 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.

25.10 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 25-15 shows the algorithm because some cases are not covered by RFC 1549.

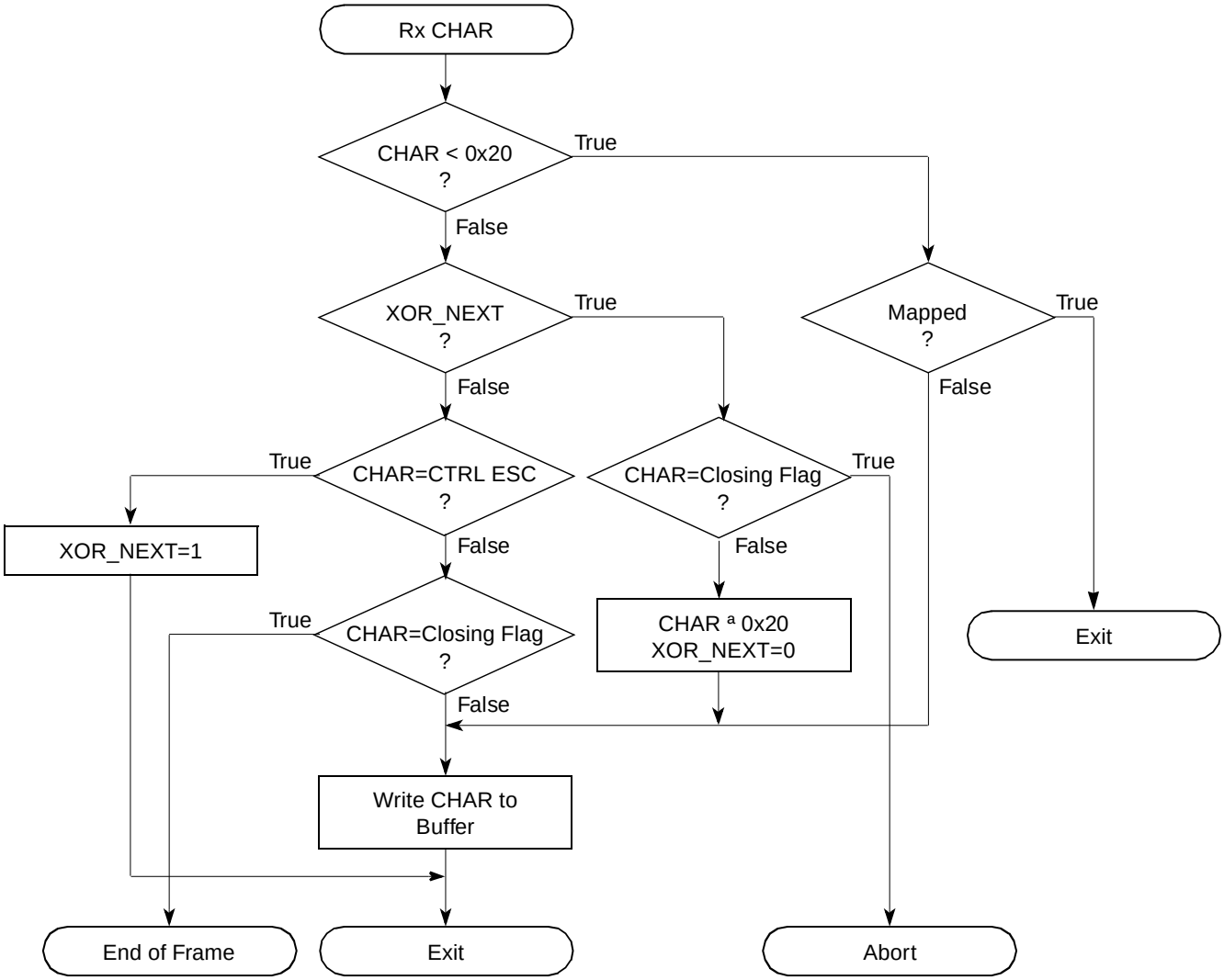


Figure 25-15. Receive Flowchart

25.11 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.

25.12 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.

25.13 Asynchronous HDLC Mode Parameter RAM

For asynchronous HDLC mode, the protocol-specific area of the UCC parameter RAM is mapped as in Table 25-16.

Table 25-16. 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 25-16. 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 25-16. 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 25-16 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 25-16. TXCTL_TBL/RXCTL_TBL

25.14 Configuring GUMR and UDSR for Asynchronous HDLC

General UCC parameters can be configured as described in Chapter 24, “UCC as Slow Communications Controllers” except for the following changes to the general UCC mode register and the UCC data synchronization register.

25.14.1 General UCC Mode Register (GUMR)

Table 25-17 shows asynchronous HDLC-specific information for the GUMR.

Table 25-17. 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.

25.14.2 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.

25.15 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 ($\overline{CD}$ and $\overline{CTS}$) is reported through the parallel I/O pins; a maskable interrupt is generated when the status of either line changes.

25.16 Asynchronous HDLC Commands

The transmit and receive commands are issued to the RISC command register (CECR).

Transmit commands are described in Table 25-18. 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 25-18. 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	Reenables 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 25-19 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 25-19. 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.

25.17 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 25-20 describes transmit errors.

Table 25-20. 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 25-21 describes reception errors.

Table 25-21. 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.

25.18 UCC Asynchronous HDLC Registers

The following sections describe the UCC registers when in asynchronous HDLC mode.

25.18.1 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 25-17, 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 W	—	GLR	GLT	—	IDL	—	BRKE	BRKS	TXE	RXF	BSY	TXB	RXB
Reset	All zeros												

Figure 25-17. Asynchronous HDLC Event Register (UCCE)/Asynchronous HDLC Mask Register (UCCM)

Table 25-22 describes UCCE/UCCM fields.

Table 25-22. 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 25-22. 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.

25.18.2 UCC Asynchronous HDLC Status Register (UCCS)

The UCC asynchronous HDLC status register (UCCS), shown in Figure 25-18, 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.

Address: Access: Read/Write

	0–2	3	4	5–6	7	8	9	10	11	12	13	14	15
R W	—	GLR	GLT	—	IDL	—	BRKE	BRKS	TXE	RXF	BSY	TXB	RXB
Reset	All zeros												

Figure 25-18. Asynchronous HDLC Event Register (UCCE)/Asynchronous HDLC Mask Register (UCCM)

Table 25-9 describes asynchronous HDLC UCCS fields.

Table 25-23. 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.

25.18.3 Asynchronous HDLC Mode Register (UPSMR)

When the UCC is in asynchronous HDLC mode, the UPSMR, shown in Figure 25-19, acts as the asynchronous HDLC mode register.

Address: Access: Read/Write

	0	1	2–3	4–15
R W	FLC	—	CHLN	—
Reset	All zeros			

Figure 25-19. Asynchronous HDLC Mode Register (UPSMR)

Table 25-24 describes UPSMR fields.

Table 25-24. 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.

25.19 UCC Asynchronous HDLC RxBDs

The QUICC Engine module uses the RxBD, shown in Figure 25-20, to report on received data.

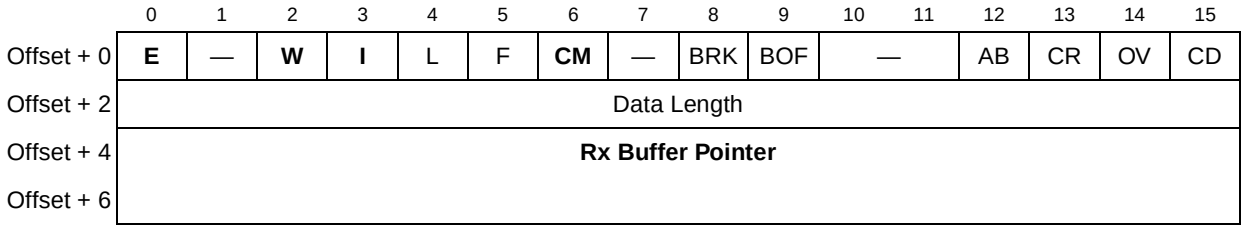


Figure 25-20. UCC Asynchronous HDLC RxBDs

Table 25-25 describes the UCC asynchronous HDLC RxBD status and control fields.

Table 25-25. 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 25-25. 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.

25.20 UCC Asynchronous HDLC TxBDs

The QUICC Engine module uses the TxBD, shown in Figure 25-21, to confirm transmissions and indicate error conditions.

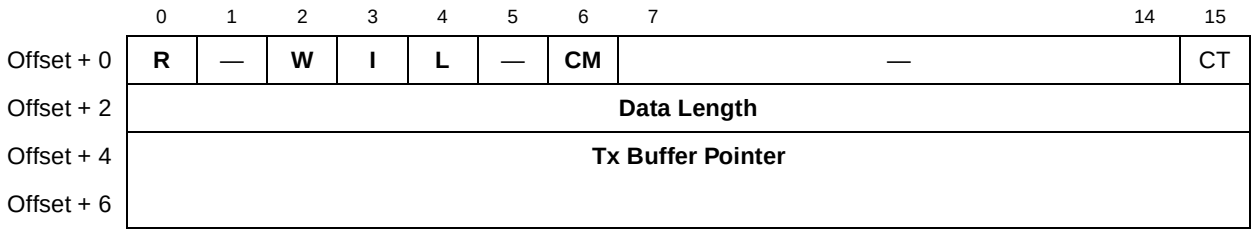


Figure 25-21. UCC Asynchronous HDLC TxBDs

Table 25-26 describes the UCC asynchronous HDLC TxBD status and control fields.

Table 25-26. 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.

25.21 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.

25.22 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 25-27 and Table 25-28 present the various parameters that impact the multi-user RAM usage.

Table 25-27. Tx and Rx Parameter RAM Usage

Data structure	Size [bytes]	Multiply by
Global Parameter RAM	256	1

Table 25-28. 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 25-29 presents the overall usage of multi-user RAM.

Table 25-29. 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 26
UCC BISYNC Mode

26.1 Introduction

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 26-1. 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 26-1. 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 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.

1.The transparent frame in BISYNC is not related to the transparent mode, discussed in Chapter 28, “Transparent Controller”

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.

26.1.1 BISYNC Block Diagram

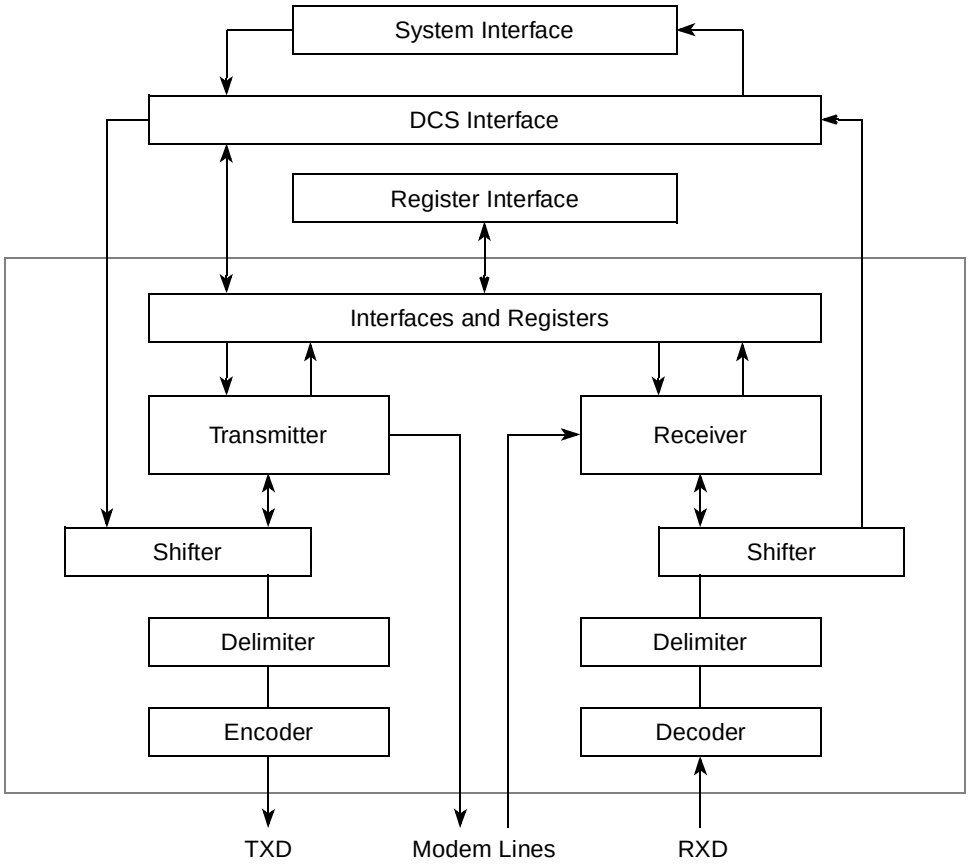


Figure 26-2. BISYNC Block Diagram

26.1.2 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

26.1.3 Modes of Operation

The UCC as a BISYNC controller works with two major modes:

- Transparent mode
- Nontransparent mode

26.2 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 26-1 below:

Table 26-1. 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

26.2.1 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 26-2. 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).
$\overline{\text{CTS}}$	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.

Table 26-2. Interface A—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{RTS}}$	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.

26.3 Memory Map/Register Definition

26.3.1 Overview

Table 26-3. UCC BISYNC Register Summary

Offset [1]	Register	Access	Reset Value	Section/Page
0x0	General UCC Mode Register (GUEMR)	R/W	0x0000_0000	24.2.2/24-2
0x8	UPSMR BISYNC Mode Register	R/W	0x0000_0000	26.3.2.5/26-9
0xC	UCC Data Synchronization Register (UDSR)	R/W	0x7E_7E	24.2.3/24-7
0x10	UCCE UCC BISYNC Event Register	R/W	0x0000_0000	26.3.2.8/26-14
0x14	UCCM UCC BISYNC Mask Register	R/W	0x0000_0000	26.3.2.8/26-14

[1] From UCC*x* base address

26.3.2 Register Descriptions

26.3.2.1 UCC BISYNC Parameter RAM

For BISYNC mode, the protocol-specific area of the UCC parameter RAM is mapped as shown in Table 26-4. Fields shown in bold text must be initialized by the user.

Table 26-4. UCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**CRCC**	Word	CRC constant temp value
0x38	**PRCRC**	Hword	Preset receiver/transmitter CRC16/LRC. These values should be preset to all ones or zeros, depending on the BCS used.
0x3A	**PTCRC**	Hword	
0x3C	**PAREC**	Hword	Receive parity error counter. This 16-bit (modulo 2^{16}) counter maintained by the QUICC Engine module counts parity errors on receive if the parity feature of BISYNC is enabled. Initialize PAREC while the channel is disabled.

Table 26-4. UCC BISYNC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
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 26.3.2.3, "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 26.3.2.4, "UCC BISYNC DLE Register (BDLE)".
0x42	**CHARACTER1**	Hword	Control character 1–8. These values represent control characters that the BISYNC controller recognizes. See Section 26.3.2.2, "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 26.3.2.2, "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.

[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 24.2.3, "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.

26.3.2.2 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 26-3.

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 26-3. Control Character Table and RCCM

[1] From UCC*x* page base address

Table 26-5 describes control character table and RCCM fields.

Table 26-5. 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.

1 From UCC*x* page base address

26.3.2.3 BISYNC SYNC Register (BSYNC)

BSYNC, shown in Figure 26-4, 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 26-4. BISYNC SYNC (BSYNC)

1 From UCC*x* page base address

Table 26-6 describes BSYNC fields.

Table 26-6. 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

26.3.2.4 UCC BISYNC DLE Register (BDLE)

BDLE, shown in Figure 26-5, 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 26-5. BISYNC DLE (BDLE)

1 From UCC*x* page base address

Table 26-7 describes the BDLE fields.

Table 26-7. 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 26-7. BDLE Field Descriptions (continued)

Bits	Name	Description
2–7	—	All zeros
8–15	DLE	DLE character

26.3.2.4.1 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 26-8. 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 26-8. 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															

26.3.2.5 BISYNC Mode Register (UPSMR)

UPSMR is shown in Figure 26-6. 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 26-6. Protocol-Specific Mode Register for BISYNC (UPSMR)

Table 26-9 describes UPSMR fields.

Table 26-9. 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 26-9. 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)

26.3.2.6 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 26-7 shows the UCC BISYNC RxBD.

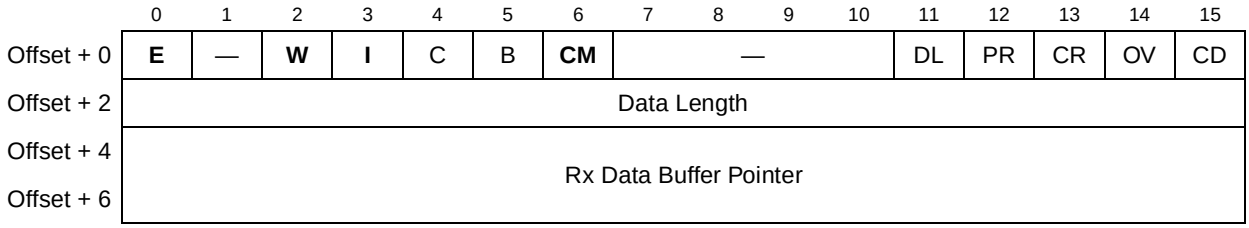


Figure 26-7. UCC BISYNC RxBD

Table 26-10 describes UCC BISYNC RxBD status and control fields. Fields shown in bold type must be initialized by the user.

Table 26-10. 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 24.3, "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.

26.3.2.7 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.

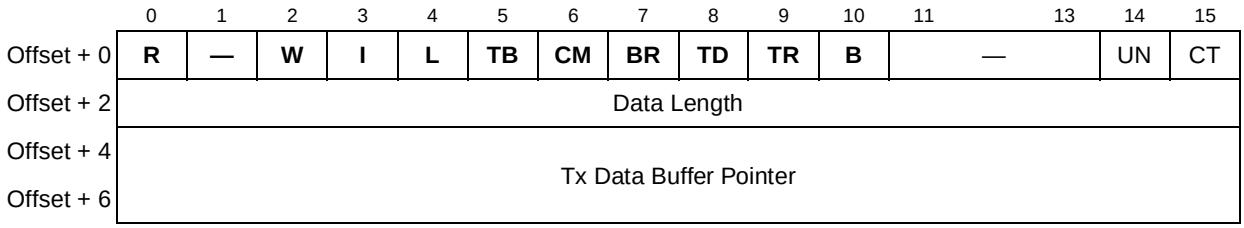


Figure 26-8. UCC BISYNC Transmit BD (TxBD)

Table 26-11 describes UCC BISYNC TxBD status and control fields. Fields shown in bold type must be initialized by the user.

Table 26-11. 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 26-11. 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 24.3, “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.

26.3.2.8 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 26-9. 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

	0–7	8	9	10	11	12	13	14	15
R W	—	GRA	—		TXE	RCH	BSY	TXB	RXB
Reset	All zeros								

Figure 26-9. BISYNC Event Register (UCCE)/BISYNC Mask Register (UCCM)

Table 26-12 describes UCCE and UCCM fields.

Table 26-12. 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.

26.4 Functional Description

26.4.1 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.

26.4.2 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 26.3.2.2, "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).

26.4.3 UCC BISYNC Commands

Transmit and receive commands are issued to the QUICC Engine module command register (CPCR). Transmit commands are described in Table 26-13.

Table 26-13. 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 26-13. 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 26-14.

Table 26-14. 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.

26.4.4 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 26-15 describes transmit errors.

Table 26-15. 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 26-16 describes receive errors.

Table 26-16. 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.

26.5 Initialization Information

26.5.1 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 26-17, the control character table should be set to recognize the end of the block.

Table 26-17. 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 27
UCC for Fast Protocols

NOTE

Read Chapter 23, "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, high-speed serials (Ethernet, HDLC, HDLC bus, Transparent), and Ethernet for the first Mile (EFM). 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, POS-PHY L2 at 50 MHz (assuming QUICC Engine frequency above 100 MHz[1]).
  - MII/RMII (assuming QUICC Engine frequency above 100 MHz).
  - GMII/RGMII/TBI/RTBI (assuming QUICC Engine frequency above 250 MHz).
  - 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 400 -MHz QE clock, the UCC supports the following:
  - Full 10/100 Mbps, full-duplex/half-duplex Ethernet/IEEE 802.3x/VLAN through MII/RMII
  - Full 1000 Mbps, full-duplex Ethernet/IEEE 802.3x/VLAN through GMII/RGMII
  - Full OC-12 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.

1. Note that this is the minimal QE frequency for H/W operation. The actual QE frequency to meet the protocol performance target will be higher.
2. The QE frequency must be higher then the data bit-rate multiplied by 8/3.
3. The QE frequency must be higher then the data bit-rate multiplied by 24 (Rev1.x) or by 16 (Rev2.x).

- Programmable size-virtual FIFO buffers
- 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

27.1 Overview

The UCC protocol is chosen by programming the mode bits in the GUMR register (see Section 27.4.2.1, "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 27-1.

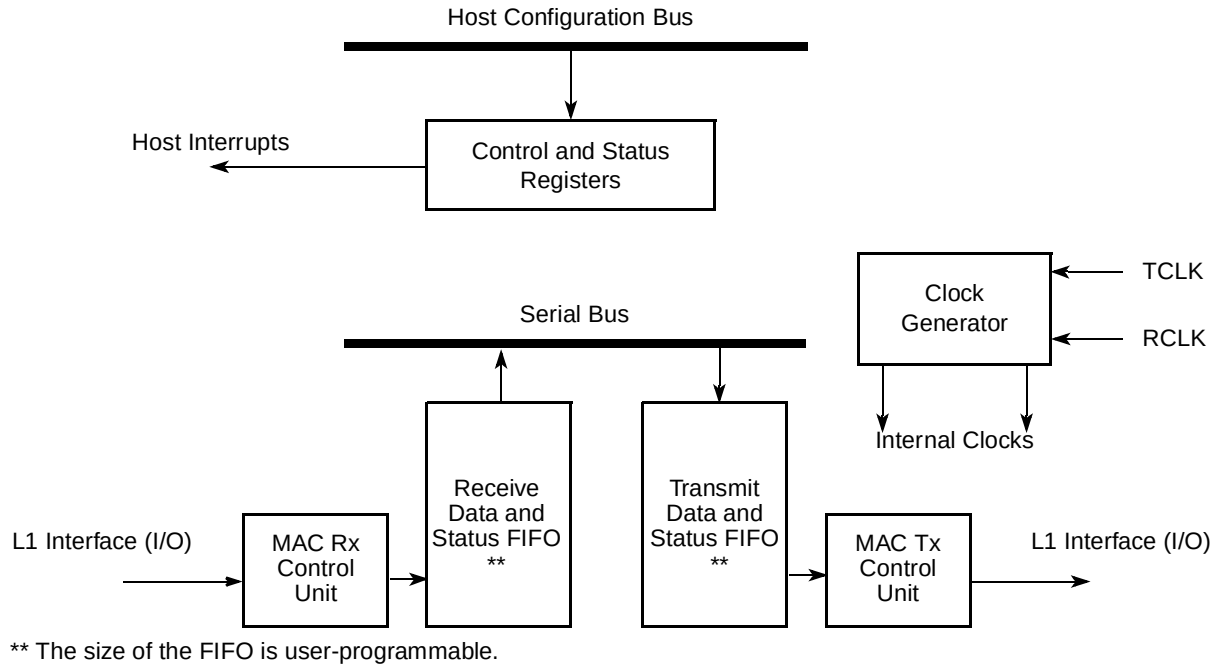


Figure 27-1. UCC Block Diagram

27.2 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 27-1 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 27.5, "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

27.3 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 23.3.2, "General UCC Extended Mode Register (GUEMR)."

The UCC Parameter RAM base addresses are programmable using the CECR command register, see Section 20.3.1, "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, and UCC5–UCC8 are backward-compatible to SCC1–SCC4 in their parameter RAM base address.

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 23.3.2, "General UCC Extended Mode Register (GUEMR)"), but the user may relocate the pages to allow improved optimization of the memory space.

27.4 Programming Model

The following sections discuss the programming model.

27.4.1 UCC Memory Map for Fast Protocols

Table 27-1. 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

Table 27-1. UCC Memory Map (Fast Mode) (continued)

Address[1]	Use	Size	Access
0x0C	UCC data synchronization register (UDSR)	2	R/W
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	

1 Offset from UCCx base

2 In Ethernet RMII mode include also Loopback and CLK that are configured in the GUMR, see Section 27.4.2.1, “GUMR in Fast Mode.”

3 Protocol specific register, described in the specific protocols’ chapters.

4 See Chapter 23, “Unified Communications Controllers (UCCs).”

27.4.2 UCC Registers in Fast Mode

The following sections discuss UCC registers in fast mode.

27.4.2.1 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 27-2. General UCC Mode Register (Fast Protocols)

Table 27-2. 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 23.5.1, "Synchronous Protocols" and in Section 23.5.2, "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 27-2. 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 (transparent mode only) 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.
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 since 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 27-2. 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 27-2. 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 1111 Reserved

27.4.3 UCC Transmit Polling Timer (UTPT)

Refer to Section 23.3.3, “UCC Transmit Polling Timer (UTPT).”

27.4.4 UCC Transmit On Demand Register (UTODR)

Refer to Section 23.3.4, “UCC Transmit-On-Demand Register (UTODR).”

27.4.5 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 27-3. UCC Data Synchronization Register (Fast Protocols)

27.4.6 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 27-4 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 27-4. Bus Mode Registers (RBMR and TBMR)

Table 27-3 describes RBMR*x*/TBMR*x* fields.

Table 27-3. 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 19.1.8.2, "Serial DMA Mode Register (SDMR)."

Table 27-3. 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 Figure 19-1and Section 5.4.3.7, “Debug Configuration.”
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 19-11 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. SeeFigure 19-1 in System Interface chapter.

27.4.7 UCC Event Register (UCCE)

Refer to Section 23.3.5, “UCC Event Register (UCCE),” and to the protocol specific chapters of the user manual.

27.4.8 UCC Mask Register (UCCM)

Refer to Section 23.3.6, “UCC Mask Register (UCCM),” and to the protocol specific chapters of the user manual.

27.4.9 UCC Status Register

Refer to Section 23.3.7, “UCC Status Register,” and to the protocol specific chapters of the user manual.

27.5 Fast Protocol FIFO Configuration Registers

The following sections discuss fast protocol FIFO configuration registers.

27.5.1 Receive Virtual FIFO Base Register (URFB)

URFB configures the receive virtual FIFO base address relative to the multi-user RAM base address.

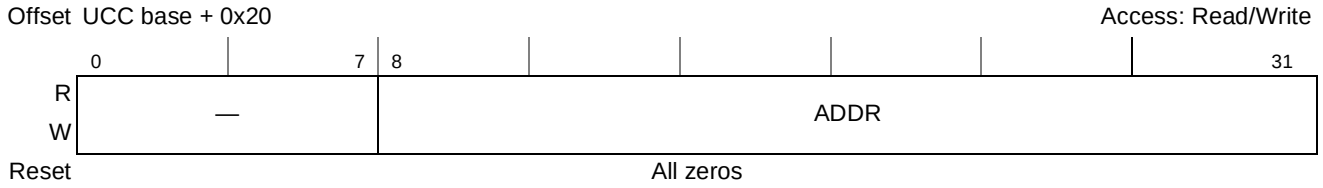


Figure 27-5. Receive Virtual FIFO Base Register (Fast Protocols)

Table 27-4. 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)

27.5.2 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.

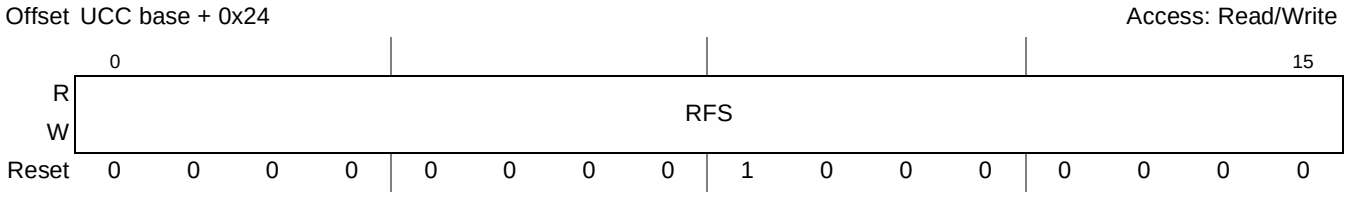


Figure 27-6. Receive Virtual FIFO Size Register (Fast Protocols)

Table 27-5. 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: • Gigabit Ethernet 2048 bytes • Fast Ethernet 512 bytes • HDLC/Transparent up to 10Mbps128 bytes • HDLC/Transparent above 10Mbps 256 bytes • ATM OC-12 Rate: 1024 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.

27.5.3 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.

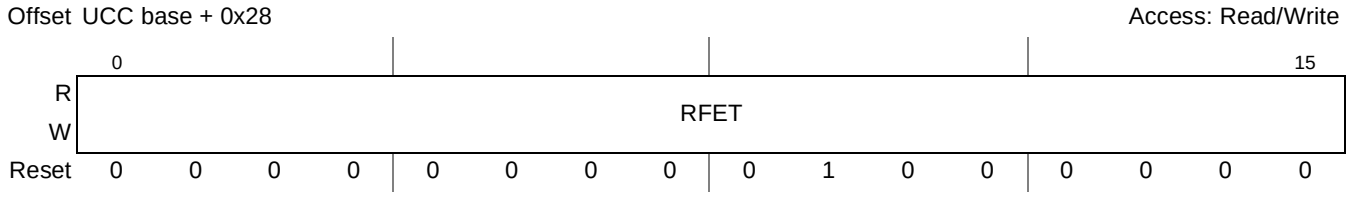


Figure 27-7. Receive Virtual FIFO Emergency Threshold (Fast Protocols)

Table 27-6. 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.

27.5.4 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, see Section 30.11.7.6, "AN Advertisement Register (ANA)." In ATM protocol, during the special emergency state the ATM transmitter may pause transmit.

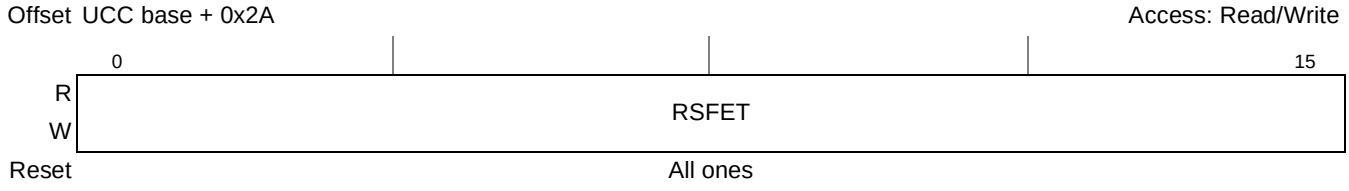


Figure 27-8. Receive Virtual FIFO Special Emergency Threshold (Fast Protocols)

Table 27-7. 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. Recommended for use in Gigabit Ethernet and ATM at rates above OC-3. 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.

27.5.5 Transmit Virtual FIFO Base Register (UTFB)

UTFB configures the transmit virtual FIFO base address relative to the multiuser RAM base address.

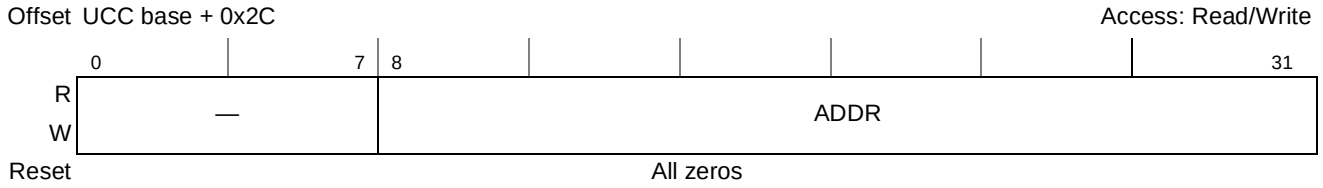


Figure 27-9. Transmit Virtual FIFO Base Register (Fast Protocols))

Table 27-8. 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).

27.5.6 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.

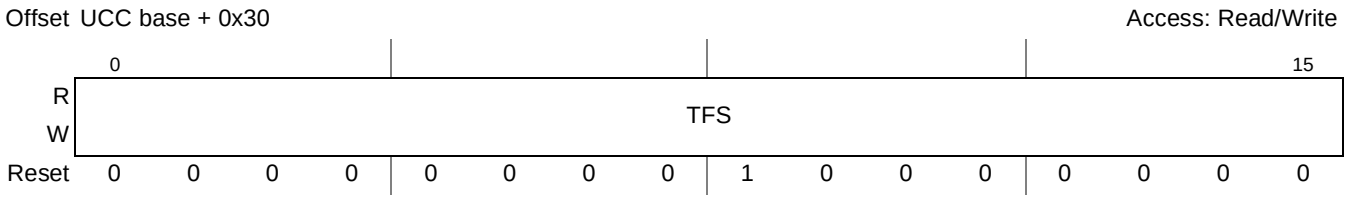


Figure 27-10. Transmit Virtual FIFO Size Register (Fast Protocols)

Table 27-9. 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: • Gigabit Ethernet 2048 bytes • Fast Ethernet 512 bytes • HDLC/Transparent up to 10Mbps128 bytes • HDLC/Transparent up to 10Mbps 256 bytes • ATM OC-12 Rate 1024 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.

27.5.7 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.

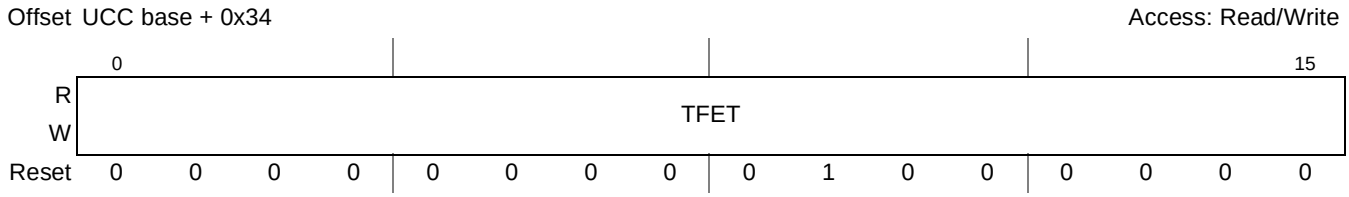


Figure 27-11. Transmit Virtual FIFO Emergency Threshold

Table 27-10. 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.

27.5.8 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.

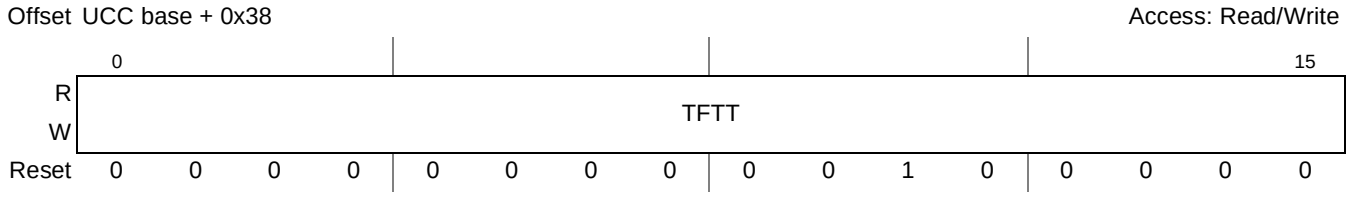


Figure 27-12. Transmit Virtual FIFO Transmit Threshold

Table 27-11. UTFTT (in Fast Mode) Register Field Descriptions

Bits	Name	Description
0–15	TFTT[1]	Transmit virtual FIFO transmit threshold (in bytes). The recommended value (for most applications) is 0x40 bytes, or up to 1/4 of the FIFO size. This threshold represents the amount of bytes that should be in virtual FIFO before transmission starts (transmission will start before reaching that watermark if the entire frame is in the virtual FIFO).

[1] Must be 8 aligned to 8 bytes.

27.5.9 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).

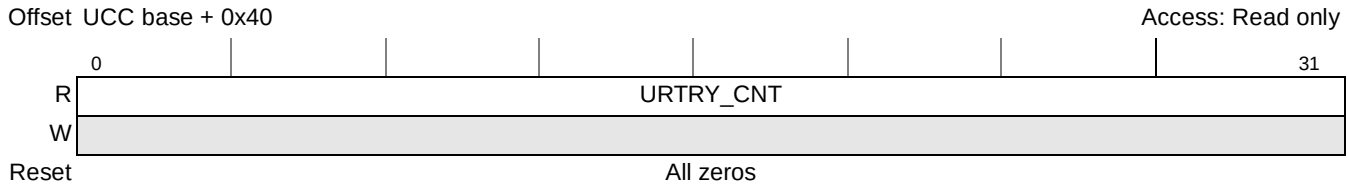


Figure 27-13. DCS Retry Counter Register (URTRY)

Table 27-12. URTRY Register Field Descriptions

Bits	Name	Description
0–31	URTRY_CNT	The number of retransmissions. Read only, cleared by any write to it.

Chapter 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

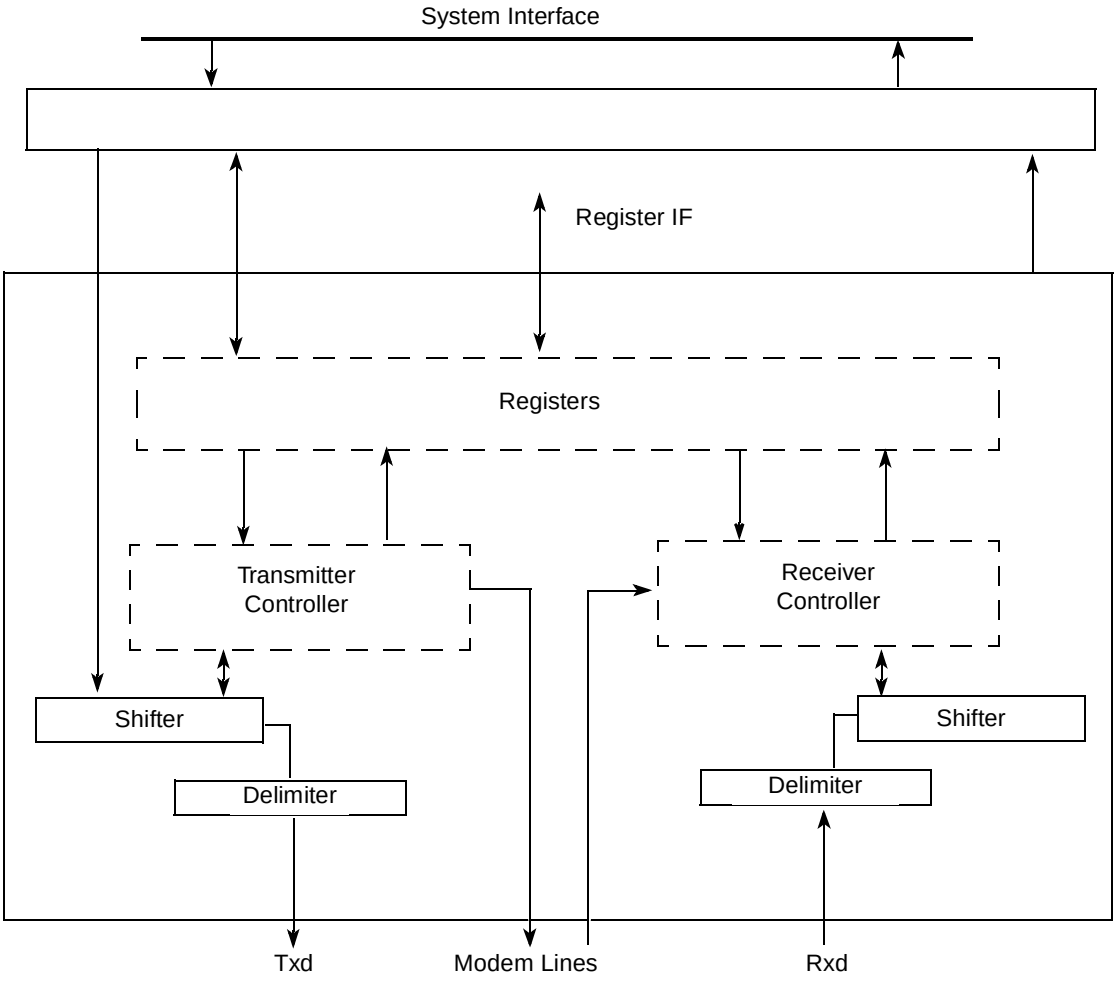


Figure 28-1. Transparent Block Diagram

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		27.4.2.1/27-6
0x4	UCC Transparent Mode Register (UPSMR)	R/W	0	28.4.2.2/28-6
0x8	Transmit On Demand Register (UTODR)	R/W		23.3.4/23-7
0xC	UCC Data Synchronization Register (UDSR)	R/W		27.4.5/27-11
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 27.4.6, “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 27.4.6, “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 29.2.2.2, “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.

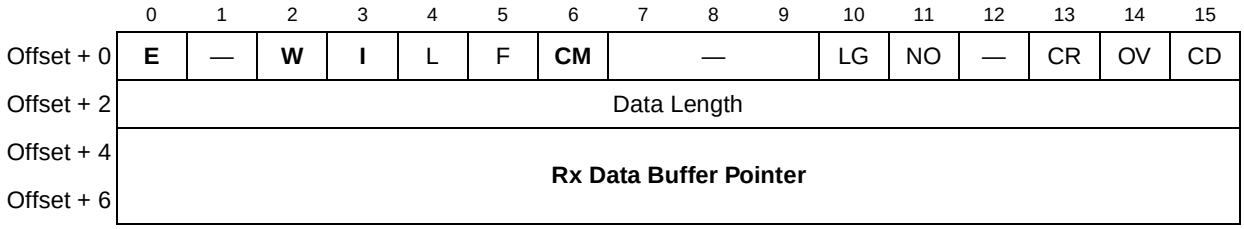


Figure 28-3. UCC Transparent Receive Buffer Descriptor (RxBD)

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.

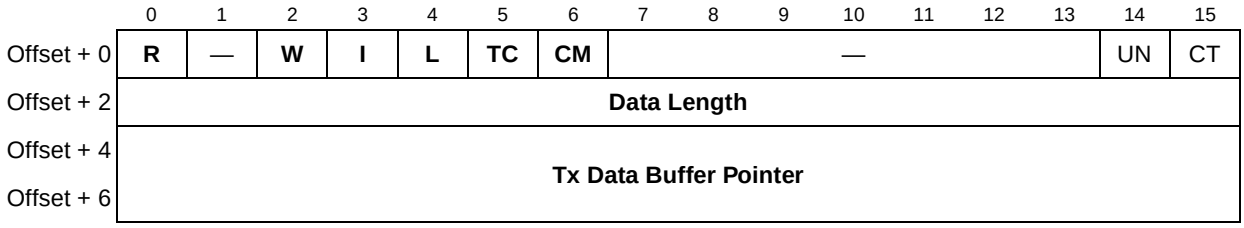


Figure 28-4. UCC Transparent Transmit Buffer Descriptor (TxBD)

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 28-7 represents the UCCE/UCCM.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: User 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. UCC Transparent Mode Register (UPSMR)

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:
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.

msb							lsb
Valid data				1	0	0	0

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 27.4.2.1, “GUMR in Fast Mode.” The pattern is defined in the UDSR; see Section 27.4.5, “UCC Data Synchronization Register (UDSR).” GUMR[SYNL] defines the SYNC pattern length. The synchronization pattern is shown in Figure 28-6.

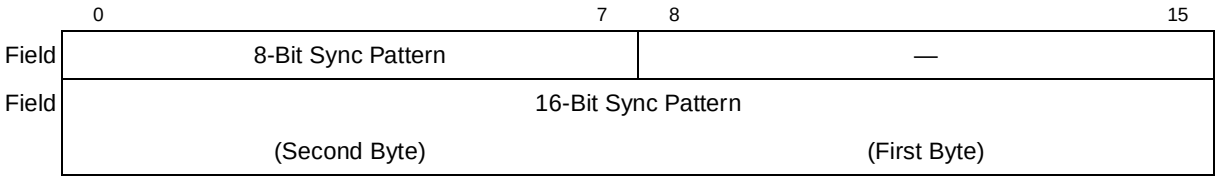


Figure 28-6. In-Line Synchronization Pattern

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

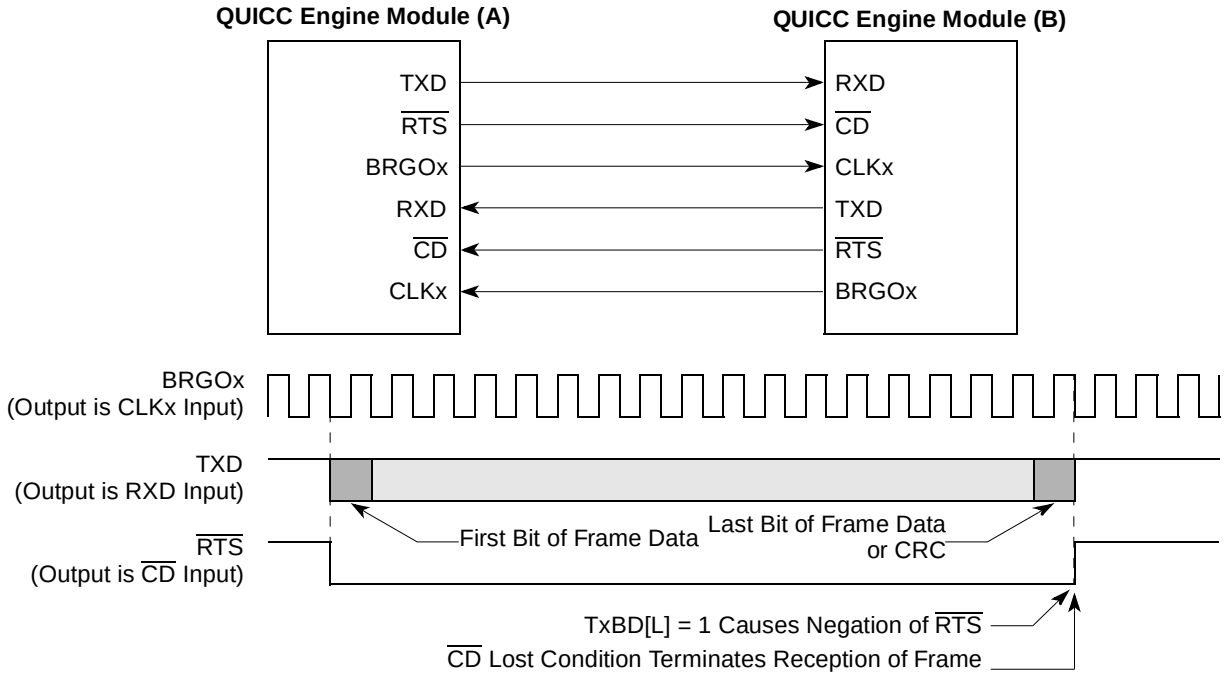
If GUMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options.

- 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 23.5, “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 since there is no defined closing flag in transparent mode. Therefore, if framing is needed, the user must use the $\overline{\text{CD}}$ line to alert the transparent controller of an end of frame.

Chapter 29
HDLC Controller

29.1 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 29-12. 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.

29.1.1 Block Diagram

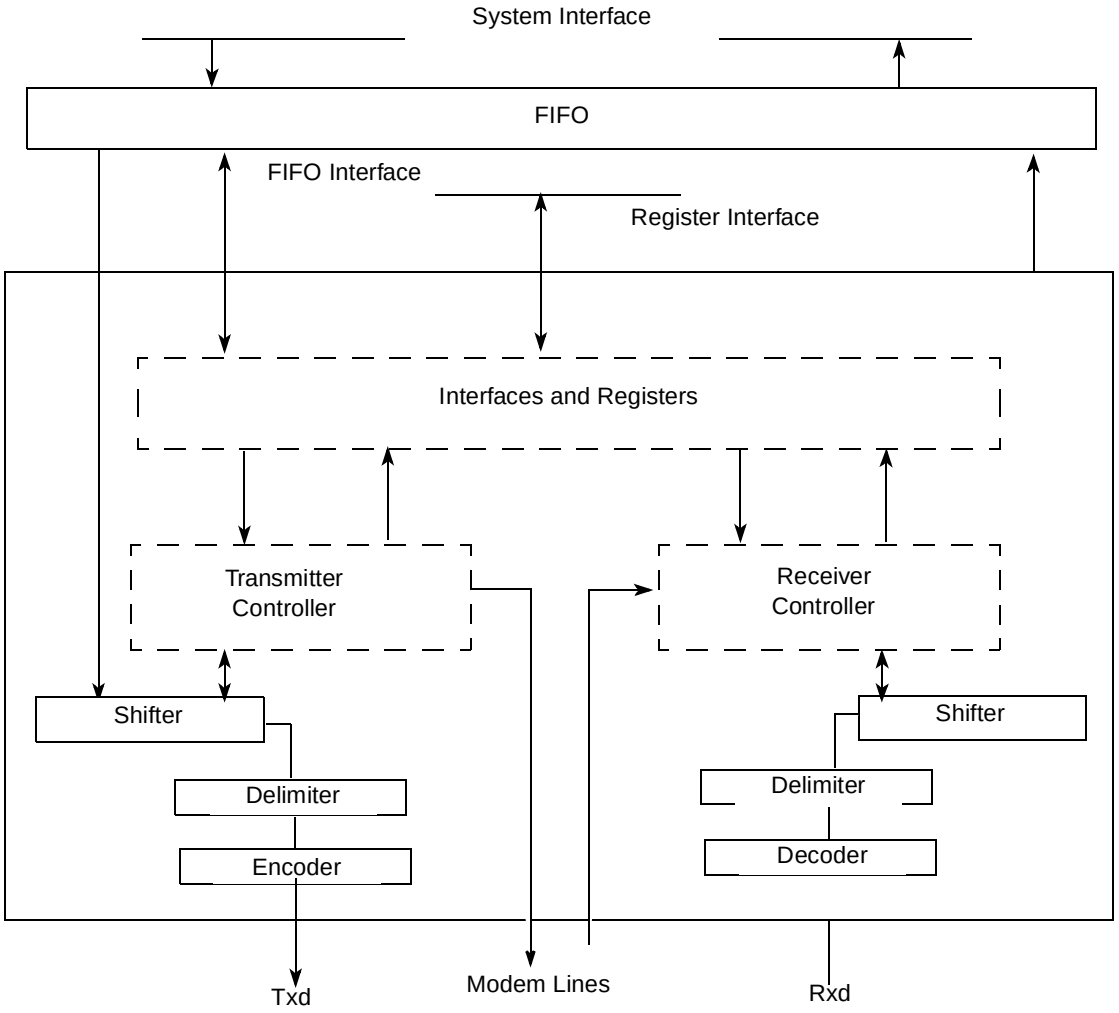


Figure 29-1. HDLC Block Diagram

29.1.2 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 70Mbps 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).

29.1.3 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.

29.2 Memory Map/Register Definition

29.2.1 Overview

Table 29-1. UCC HDLC Register Summary

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x4	UPSMR HDLC Mode Register	R/W	0x0000_0000	29.2.2.2/29-7
0x10	UCCE UCC HDLC Event Register	R/W	0x0000_0000	29.2.2.5/29-12
0x14	UCCM UCC HDLC Mask Register	R/W	0x0000_0000	29.2.2.5/29-12
0x18	UCCS UCC HDLC Status Register	R/W	0x0000_0000	29.2.2.6/29-15

29.2.2 Register Descriptions

29.2.2.1 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 29-2.

Table 29-2. 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 27.4.6, “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 27.4.6, “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 29-2. 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, $\overline{CD}$ 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 29-2. 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 29-2.
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 29-2 shows an example of using HMASK and HADDR[1–4].

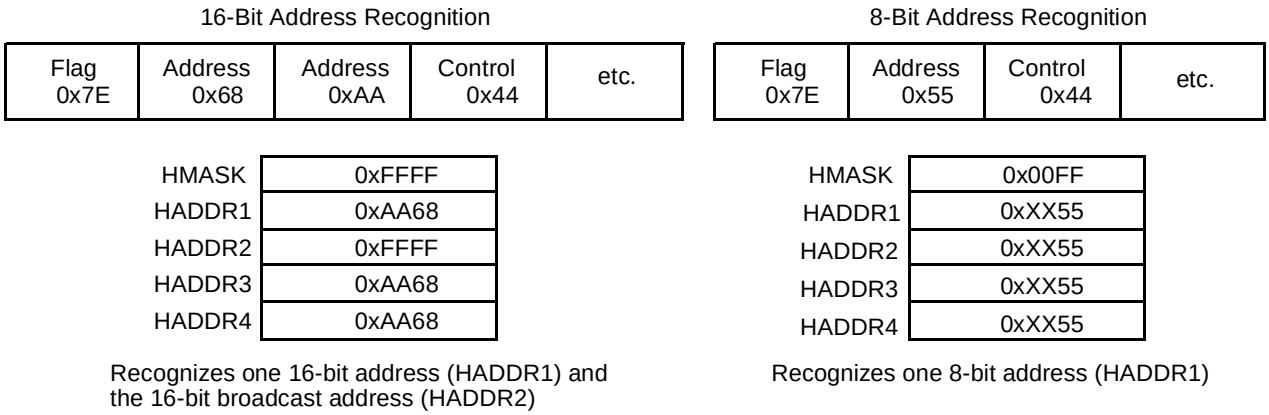


Figure 29-2. HDLC Address Recognition Example

29.2.2.2 HDLC Mode Register (UPSMR)

When a UCC is configured for HDLC mode, the UPSMR is used as the HDLC mode register, shown in Figure 29-3.

Address: UCC_base + 0x4 Access: User Read/Write

	0–3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NOF	FSE	MFF	RTE	—		TS	BUS	BRM	—	DRT	—	NBO
Reset	All zeros												

	16	17–19	20–23	24	25	26–31
R W	NBO	CW	—	CRC		—
Reset	All zeros					

Figure 29-3. HDLC Mode Register (UPSMR)

Figure 29-4. HDLC Mode Register (UPSMR)

The UPSMR fields are described in Table 29-3.

Table 29-3. 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 20.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)." Note that when UPSMR[TS] is set, the timestamp used is CETSCR2.

Table 29-3. UPSMR Field Descriptions (continued)

Bits	Name	Description
10	BUS	HDLC bus mode. 0 Normal HDLC operation. 1 HDLC bus operation is selected. See Section 29.3.3, "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.

29.2.2.3 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 29-5 shows an example of the RxBD process.

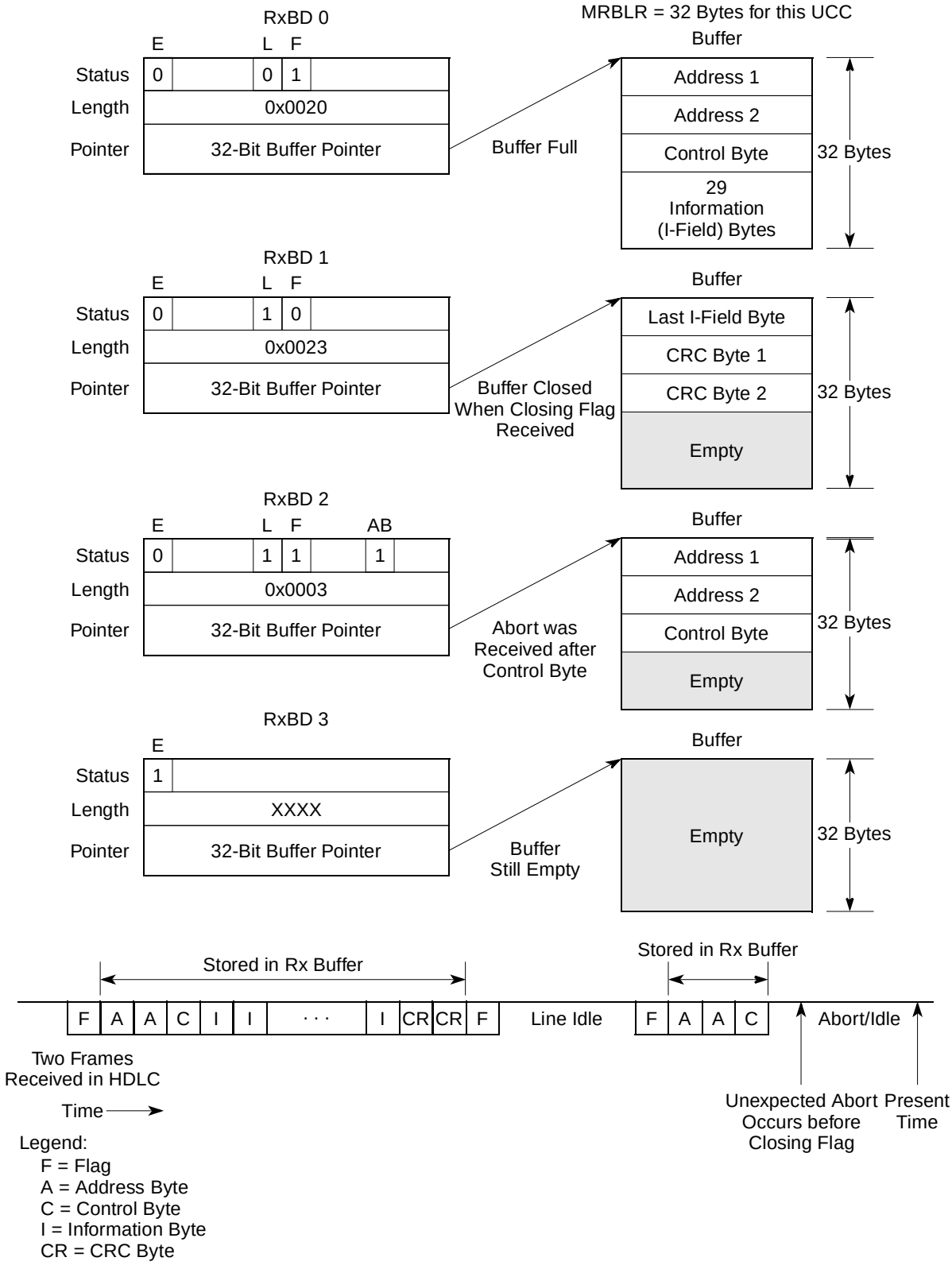


Figure 29-5. UCC HDLC Receiving Using RxBDs

Figure 29-6 shows the UCC HDLC RxBD.

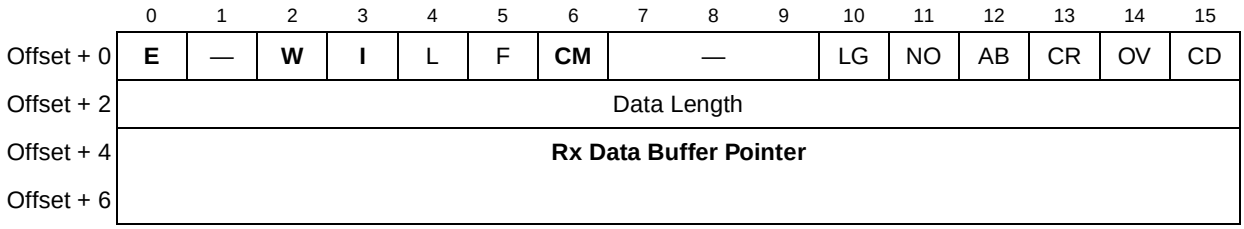


Figure 29-6. UCC HDLC Receive Buffer Descriptor (RxBD)

Table 29-4 describes RxBD fields.

Table 29-4. 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 29-4. 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 29-3).

29.2.2.4 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 29-7 shows the UCC HDLC TxBD.

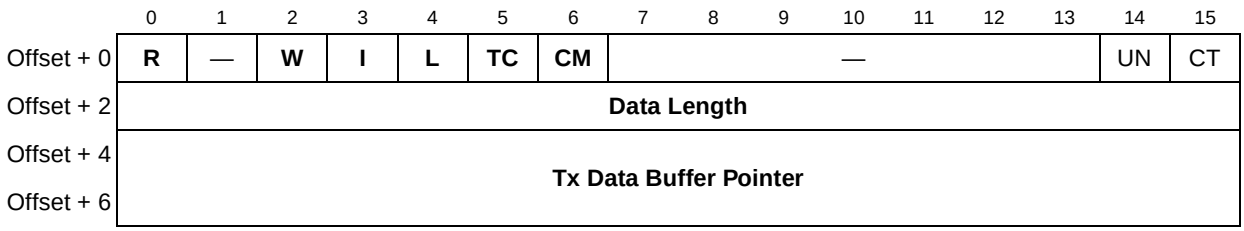


Figure 29-7. UCC HDLC Transmit Buffer Descriptor (TxBD)

Table 29-5 describes HDLC TxBD fields.

Table 29-5. 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 29-5. 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.

29.2.2.5 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 29-8 represents the UCCE/UCCM.

Address: UCC_base + 0x10 (UCCE), UCC_base + 0x14 (UCCM) Access: User Read/Write

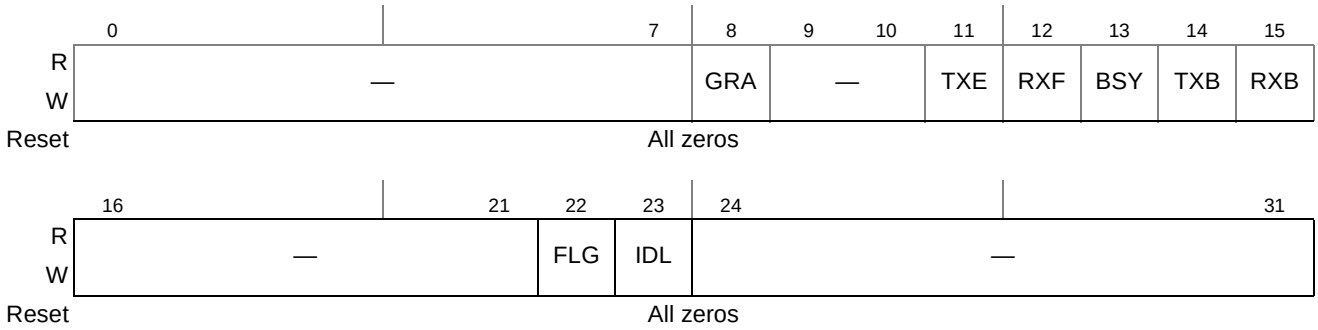


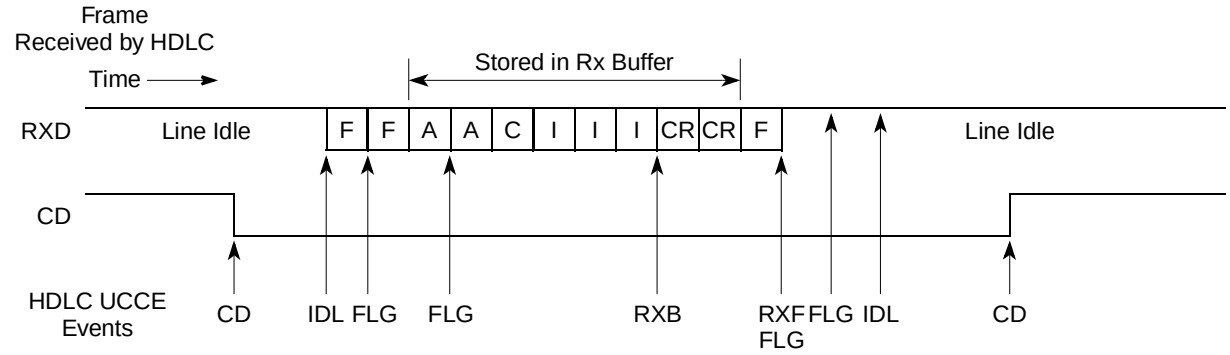
Figure 29-8. HDLC Event/Mask Register (UCCE/UCCM)

Table 29-6 describes UCCE/UCCM fields.

Table 29-6. UCCE/UCCM Field Descriptions

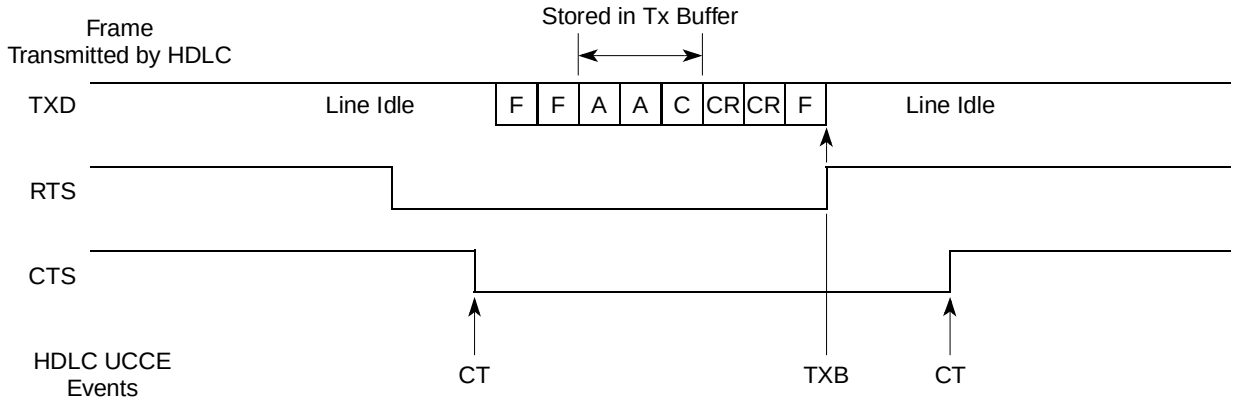
Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. GRA is set as soon as the transmitter finishes transmitting any frame that is in progress when the command was issued. It 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 29.2.2.6, “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 29.2.2.6, “UCC Status Register (UCCS).”
24–31	—	Reserved, should be cleared.

Figure 29-9 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 29-9. HDLC Interrupt Event Example

29.2.2.6 UCC Status Register (UCCS)

The UCCS register, shown in Figure 29-10, allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ signals are part of the parallel I/O port.

Address: UCC_base + 0x18 Access: Read Only

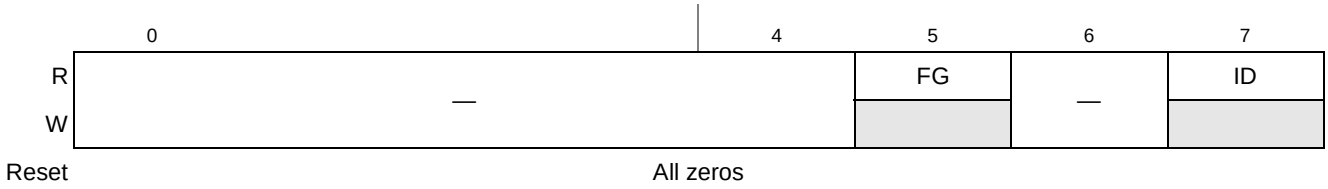


Figure 29-10. HDLC Status Register (UCCS)

Figure 29-11. HDLC Status Register (UCCS)

Table 29-7. 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.

29.3 Functional Description

29.3.1 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 29-12 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 29-12. 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.

29.3.2 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.

29.3.3 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.

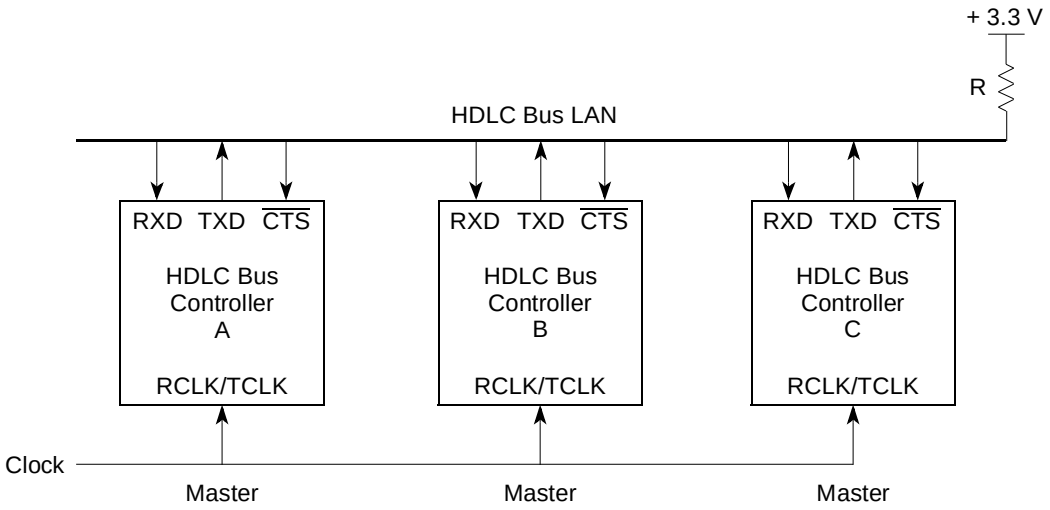
The HDLC bus differs from the I.430 and T1.605 standards as follows:

- 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 via port pin configuration or via external transceivers

Figure 29-13 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.

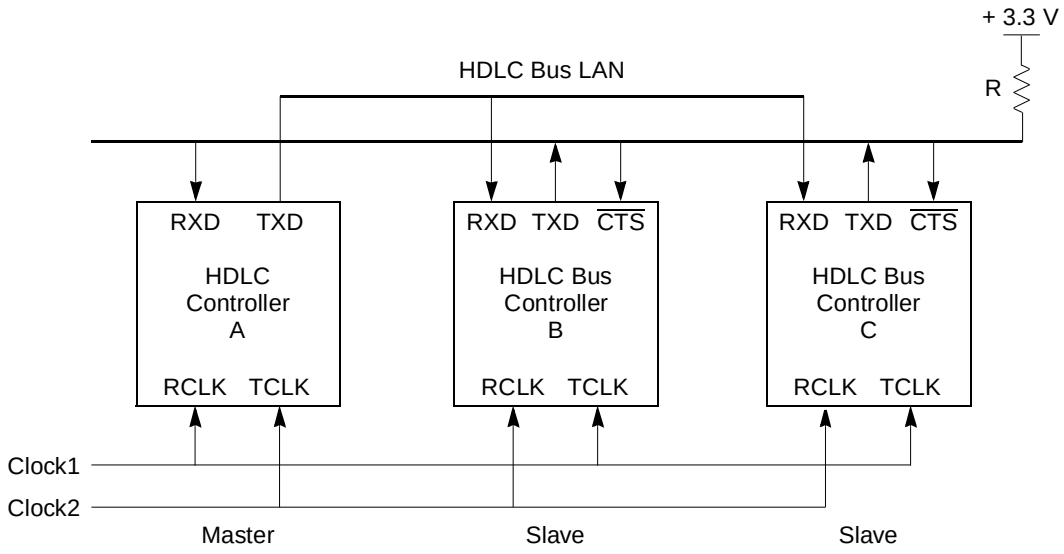


NOTES:

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 29-13. 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 29-14 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 29-14. Typical HDLC Bus Single-Master Configuration

29.3.3.1 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

29.3.3.2 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. Since the HDLC bus uses a wired-OR scheme, a transmitted zero has priority over a transmitted 1. Figure 29-15 shows how $\overline{\text{CTS}}$ is used to detect collisions.

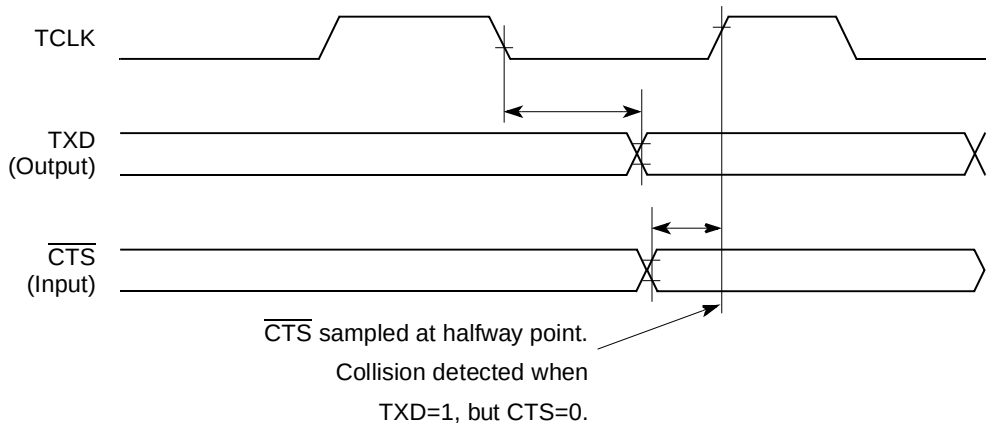


Figure 29-15. Detecting an HDLC Bus Collision

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.

29.3.3.3 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 29-16.

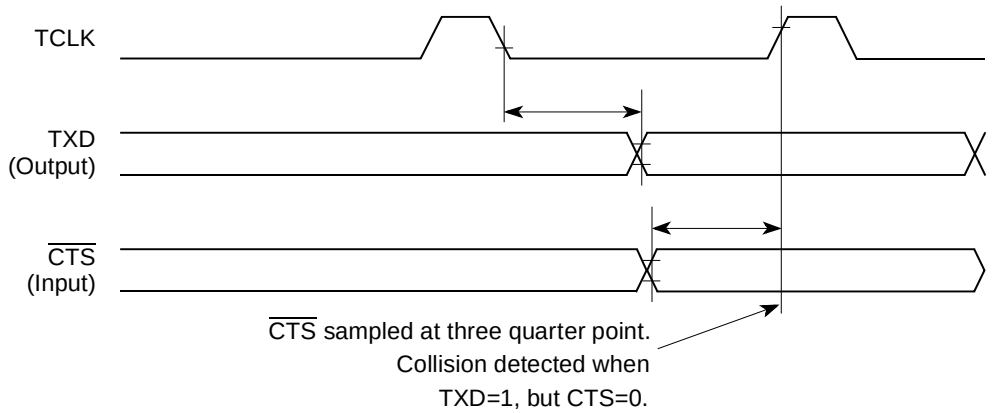
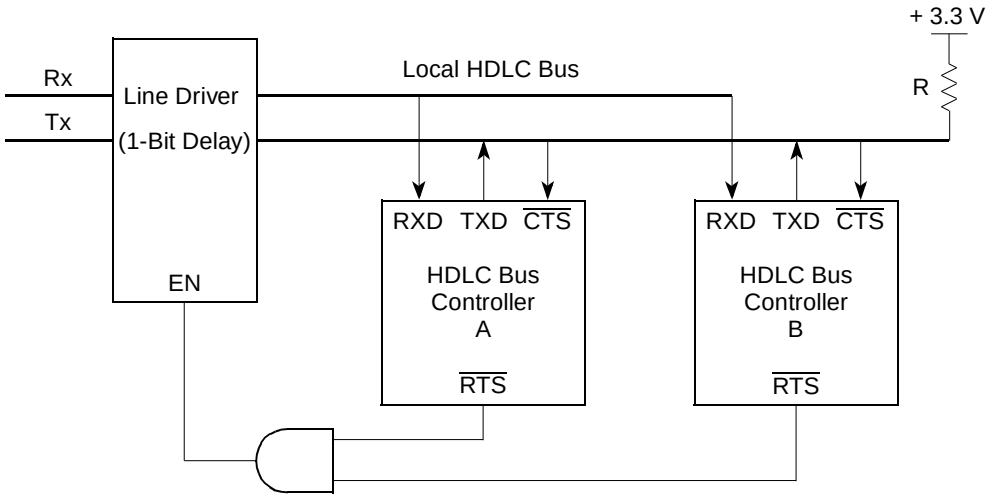


Figure 29-16. Non symmetrical Tx Clock Duty Cycle for Increased Performance

29.3.3.4 Delayed $\overline{\text{RTS}}$ Mode

Figure 29-17 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 29-17. 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 29-18 shows $\overline{\text{RTS}}$ timing.

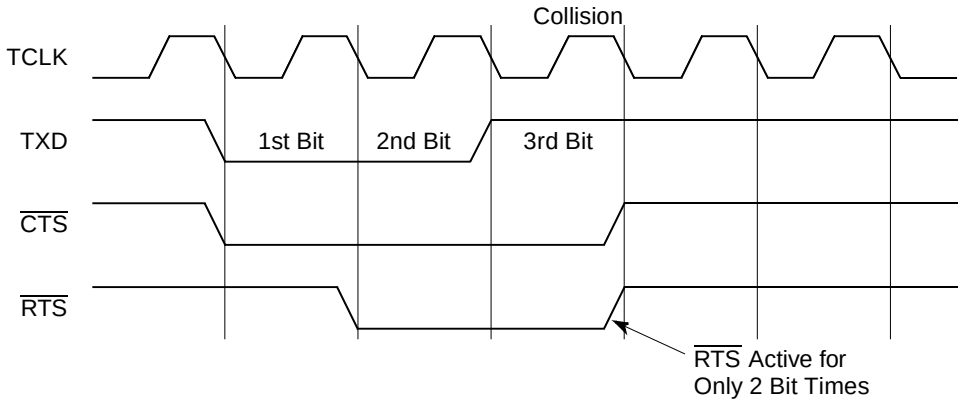
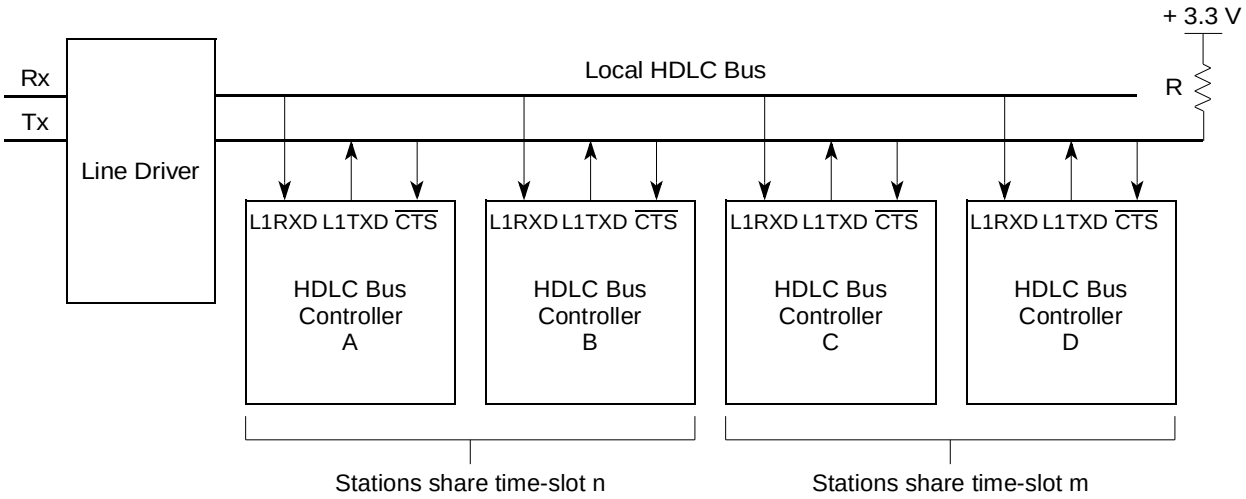


Figure 29-18. Delayed $\overline{\text{RTS}}$ Mode

29.3.3.5 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 29-19. 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 29-19. 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.

29.3.3.6 HDLC Command Set

The transmit and receive commands are issued to the CECR; see Section 20.3.1, “QUICC Engine Command Register (CECR).”

Table 29-8 describes the transmit commands that apply to the HDLC controller.

Table 29-8. 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 29-9 describes the receive commands that apply to the HDLC controller.

Table 29-9. 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.

29.3.3.7 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 29-10 describes HDLC transmission errors, which are reported through the TxBD.

Table 29-10. 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 29-11 describes HDLC reception errors, which are reported through the RxBD.

Table 29-11. 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 29-11. 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>

29.4 Initialization Information

29.4.1 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.

29.4.1.1 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.

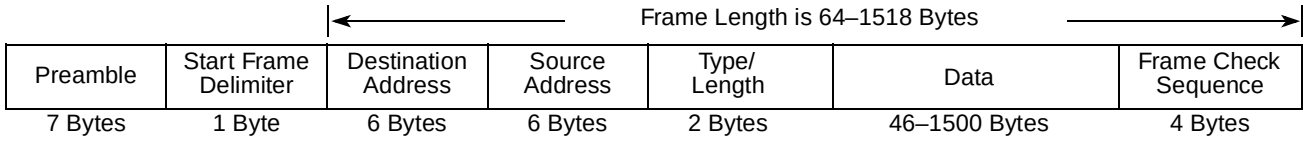
To program the general UCC mode register (GUMR), set the bits as described below:

- 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 30
UCC Ethernet Controller (UEC)

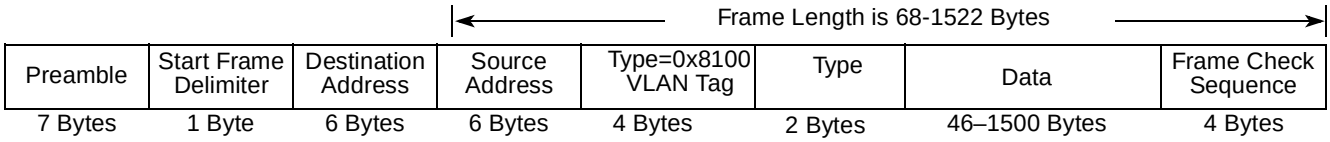
30.1 Introduction

The Ethernet IEEE Std 802.3 protocol is a widely-used LAN, based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE Std. 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 Std. 802.3 frames are based on the frame structure shown in Figure 30-1.



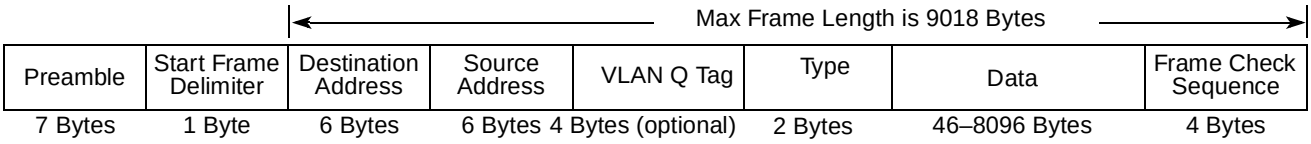
Note: The lsb of each octet is transmitted first.

Figure 30-1. Untagged Ethernet Frame Structure



Note: The lsb of each octet is transmitted first.

Figure 30-2. VLAN Tagged Ethernet Frame Structure



Note: The lsb of each octet is transmitted first.

Figure 30-3. VLAN Tagged Jumbo Ethernet Frame Structure

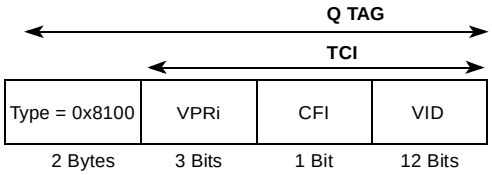


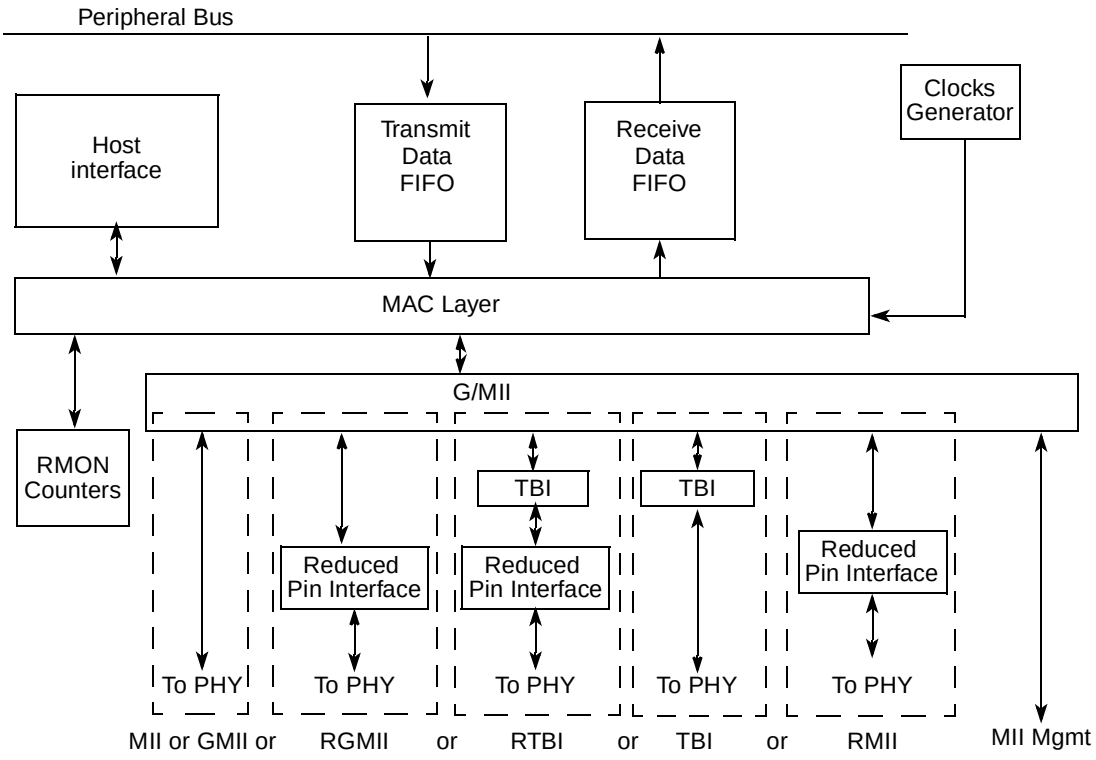
Figure 30-4. VLAN Q TAG

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 Std. 802.3 specification allowed 16-bit addressing, which has never been widely used.
- Ethernet type field/IEEE Std. 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 30-5 describes the block diagram of the UCC Ethernet controller.



Note: The TBI and RTBI are not available on all UCCs

Figure 30-5. UCC Ethernet Controller Block Diagram

30.2 Overview

The UCC Ethernet Controller is a 10/100/Gigabit 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 standard)

- RMII interface
- 1000 Mbps GMII interface (IEEE 802.3 -2002standard)
- 1000 Mbps TBI interface (IEEE 802.302002 standard)
- 1000 Mbps RGMII/RTBI reduced interface

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).

The GMII/MII provides a standard interface between the MAC layer and the physical layer for 10/100/1000 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 TBI provides connectivity between a 1000 Mbps MAC and 1000BASE-X SERDES.

30.3 Features

The UCC Ethernet Controller includes the following features:

- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE Std. 802.3 MII
  - 10/100 Mbps RMII (Consortium standard)
  - 1000 Mbps IEEE Std. 802.3-2002 GMII
  - 1000 Mbps IEEE Std. 802.3-2002 TBI
  - 1000 Mbps only supports full-duplex RTBI
  - 10/100/1000 Mbps RGMII reduced interface
- 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 (LossLess Flow Control)
- 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 Std. 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
  - Nonoctet 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 Std. 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 Std. 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
- Support for IEEE Std. 1588

30.4 Functional Description

30.4.1 Ethernet Frame Transmission

30.4.1.1 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. As a result the UEC 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 by the QUICCEngine. The user may program the polling time in UPTP register in the UCC. The deafult polling time is 256 transmit clocks. If TxBD[R] is set, the UCC Ethernet Controller begins moving 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 (TBI/RTBI/GMII/RGMII/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 is designed to schedule up to eight queues for transmission. The scheduler combines both strict priority queues (SPQ) and weighted fair queues (WFQ). This allows the user to map latency-sensitive data transmissions to the strict priority queues, while bandwidth dependent data transmissions are mapped to the weighted fair queues. In addition, the scheduler maintains a rate limiter. The rate limiter allows limiting the average bit rate transmitted by the Gigabit Ethernet Port.

If the user has a frame ready to transmit, setting 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 once 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 30-1. 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 30-1. 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 UCC Ethernet Controller (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 MAC is currently 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 completing the current transmit frame. 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 it is advised to increase the pause time.

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 may program the UEC to automatically appends FCS (32-bit CRC) 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

The following table specifies the behavior of the UEC depending on the bit programming:

Table 30-2. Tx CRC and PAD Mode

TxBD[PAD/CRC]	TxBD[TC]	MACCFG2[PAD/CRC]	MACCFG2[CRE]	CRC	PAD (frames of 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.
0	0	0	0	No	No padding
0	1	0	0	User programmable per BD	No padding
1	ignored	0	0	Padding of frames which have 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. While it is in 1000 Mbps mode it sends the least significant bit of each byte first.

30.4.2 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 (i.e. Filtering). Two modes are supported: MPC82xx backward compatible frame filtering, and Extended Frame Filtering mode. The Extended Parsing supports large memory based lookup tables. 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. While it is in 1000Mbps mode it receives the least-significant bit of a byte first.

30.4.3 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.

30.4.4 Multithreading

The UEC processes frames at wirespeed at rate of up to 1Gbps rates. In order to support this rate, 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 suggested number of threads for Gigabit Ethernet is four threads. The INITETHERNET command programs the data structures needed for multithreading. See Section 30.8, “Ethernet Command Set,” for more details.

The QUICC Engine has multiple RISC Engines. The architecture supports real-time dynamic task assignment to the RISCs. This is achieved by programming the RISC Allocation Entry in the INIT RX AND TX Host Command to 0x3. It is possible for the user to assign each task to a specific RISC in a static way. This may be done by modifying the RISC Allocation Entry in the INIT RX AND TX Host Command.

30.4.5 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 27.5, “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 QE.

NOTE

For the Rx Virtual FIFO, the user MUST allocate a memory block of size URFS+8 bytes in the multiuser RAM.

NOTE

For the Tx Virtual FIFO, if more than one thread is used, the minimum size of virtual FIFO is 816 bytes.

30.4.5.1 Virtual FIFO Overun Smoother

Overrun on the receive Virtual FIFO occurs when the QE is not able to remove data from the Virtual FIFO at the rate that the data is placed into the FIFO from the line. This condition causes the Virtual FIFO to fill up beyond its capacity, so incoming frames are discarded. With the Virtual FIFO Overrun smoother, the QE removes a programmable number of frames (OV_SkipFrame) from the Virtual FIFO, when an OV condition is detected. This feature is needed for performance enhancement in the case that the QE is heavily loaded.

In order to determine the correct value for the OV_SkipFrame parameter, the user should take the following steps:

Determine the maximum sustained Rx bit rate that can be supported by the QE without Virtual FIFO overrun (OV) condition. This variable is called ‘Sustained_Rx’ (Mbps).

Determine the maximum Rx bit under burst condition (probably 1Gbps). This variable is called ‘Burst_Rx’ Mbps.

Determine the average packet size in bytes. This variable is called ‘Ave_Pkt_Size’.

The value of OV_SkipFrame is:

*OV_SkipFrame > (Burst_Rx / Sustained_Rx) * (UCC Rx Virtual FIFO size in bytes / Ave_Pkt_Size)*

As an example:

Sustained_Rx = 333Mbps

Burst_Rx = 1Gbps

Ave_Pkt_Size = 256 bytes

UCC Rx Virtual FIFO size = 2Kbytes

*OV_SkipFrame = 1Gbps/333Mbps * 2048/256 = 24*

30.4.6 Termination and Interworking Modes of Operation

The receiver of the UCC Ethernet Controller is able to perform the following functions:

- Terminate frames to a CPU queue and issue a maskable interrupt

- Forward frames to an Etherent transmit queue (in the same port, or in an other port)
- Forward frames to an AAL5 or AAL2 Tx queue
- Forward frames to an ML/MC PPP or plain PPP Tx queue
- Terminate frame to a the CPU queue after forwarding and issue a maskable interrupt

30.4.7 IP Header Checksum

The UEC may 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 chcksum 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 incomig 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.

30.4.8 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 30-3 shows the flow-control frame structure.

Table 30-3. 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

Table 30-3. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause period measured in quanta of 512 bits time most-significant octet first.
2	Extension Field		As programmed in UEMPR register. See Section 30.5.1.17, "UCC Ethernet Mac Parameter Register (UEMPR)."
40	Reserved	—	
4	FCS		Frame check sequence (CRC)

30.4.8.1 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 27.5.3, "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 QE 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 (i.e. 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

30.4.8.1.1 Loss Less 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 (i.e. 64 byte frames). Therefore N = Virtual FIFO size / (64+8) bytes.

30.4.8.2 Receiving a Flow Control Frame

When flow-control mode is enabled (MACCFG1[RFCE] 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 frame can be sent. Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

All received FC Frames are mapped to the default receive queue as programmed in TCI field in the Global Rx Parameter RAM.

30.4.8.3 Back Pressure

Full-duplex flow control is provided for in IEEE Std. 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.

30.4.9 Frame Filtering and Address Recognition

The UEC supports two address recognition modes:

- MPC82xx backward compatible filtering mode
- Programmable Parse Command Descriptor (PCD) mode

30.4.9.1 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 30-6 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..

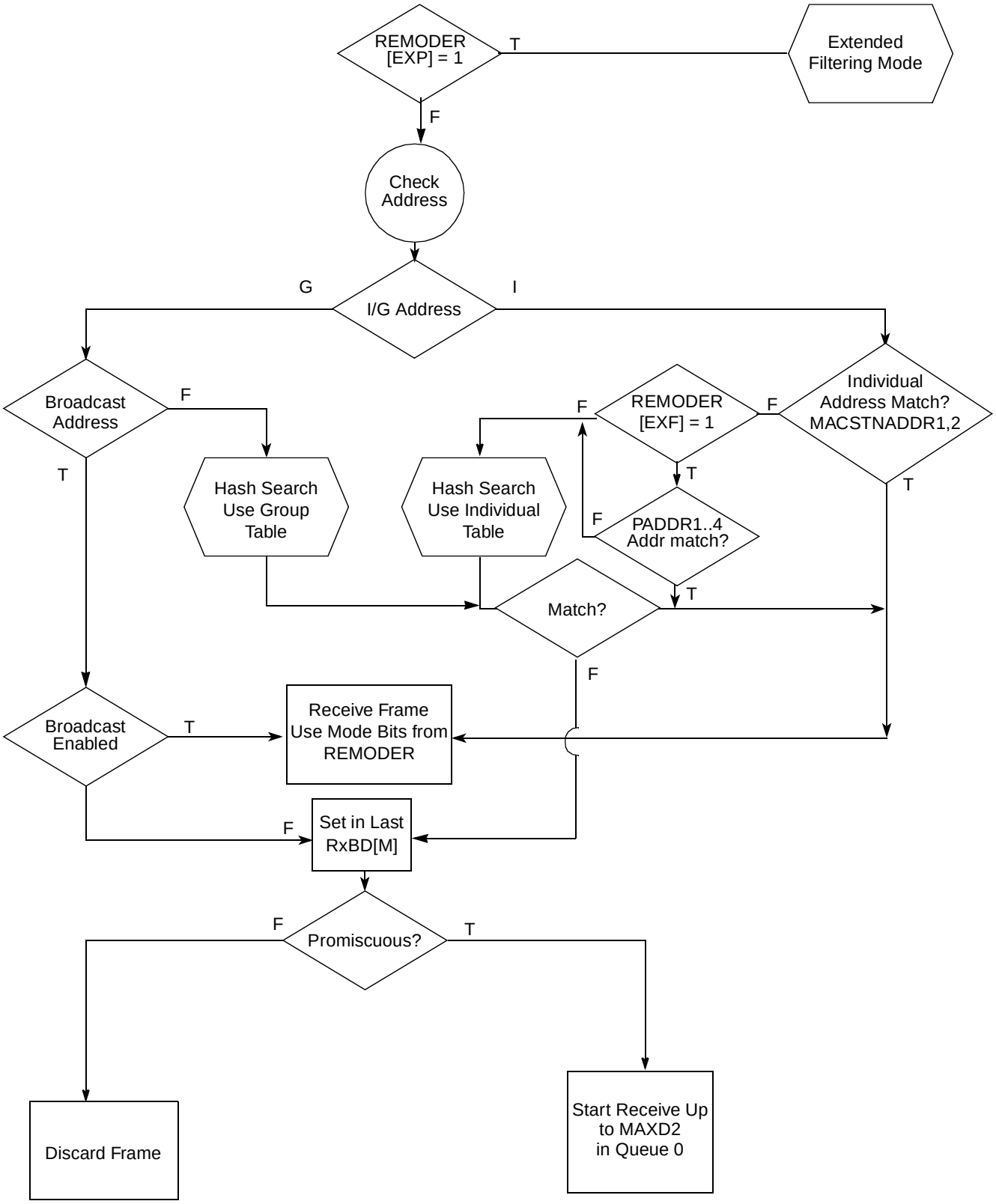


Figure 30-6. Address Filtering Flow REMODER[EXP]=0

30.4.9.1.1 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 QE 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 1 and 64. 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. In such instances, extended filtering mode is advised.

NOTE

The hash tables cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table.

30.4.9.2 Extended Parsing Mode

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. The user selects this mode 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.

30.4.9.2.1 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 16) 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 30-7 is a description of the lookup flow using this mode:

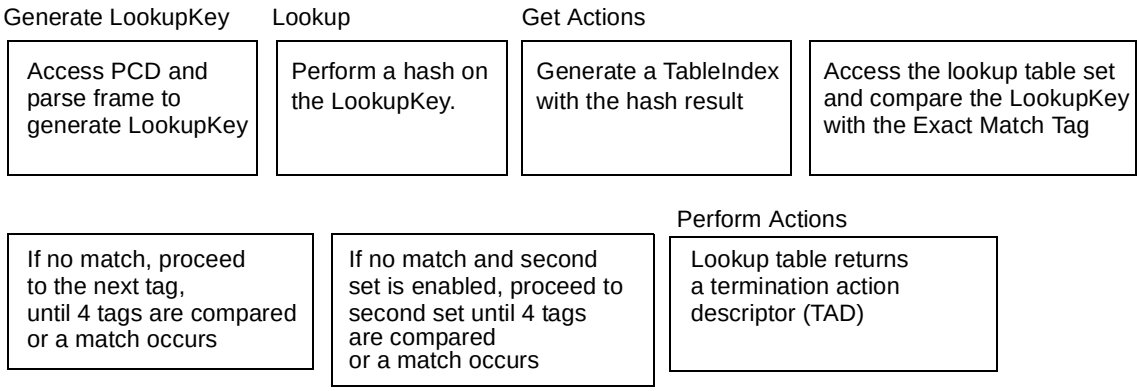


Figure 30-7. Flow for Hash mode

30.4.9.2.2 Address Filtering Flow

Figure 30-8 depicts the flow executed by the UEC in Extended Parsing mode.

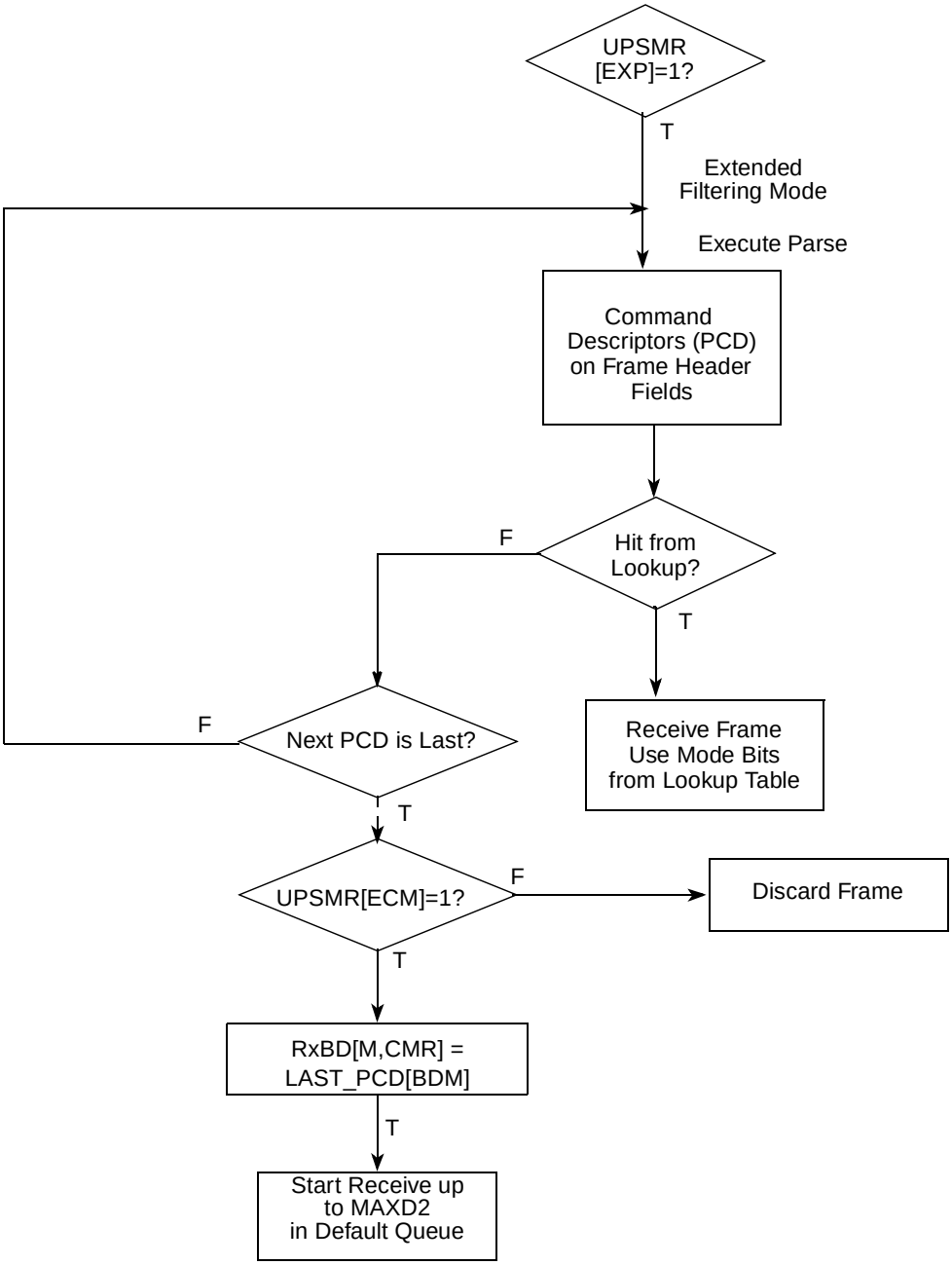


Figure 30-8. Address Filtering Flow REMODER[EXP] = 1

30.4.10 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,VNoTagOP] 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.

30.4.11 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.

30.4.12 Quality of Service (QoS)

30.4.12.1 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 (i.e. before any header manipulation peformed 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. The following table specifies how the queue priority is chosen by the UCC Ethernet Controller.

Table 30-4. 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

1 Note: 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:

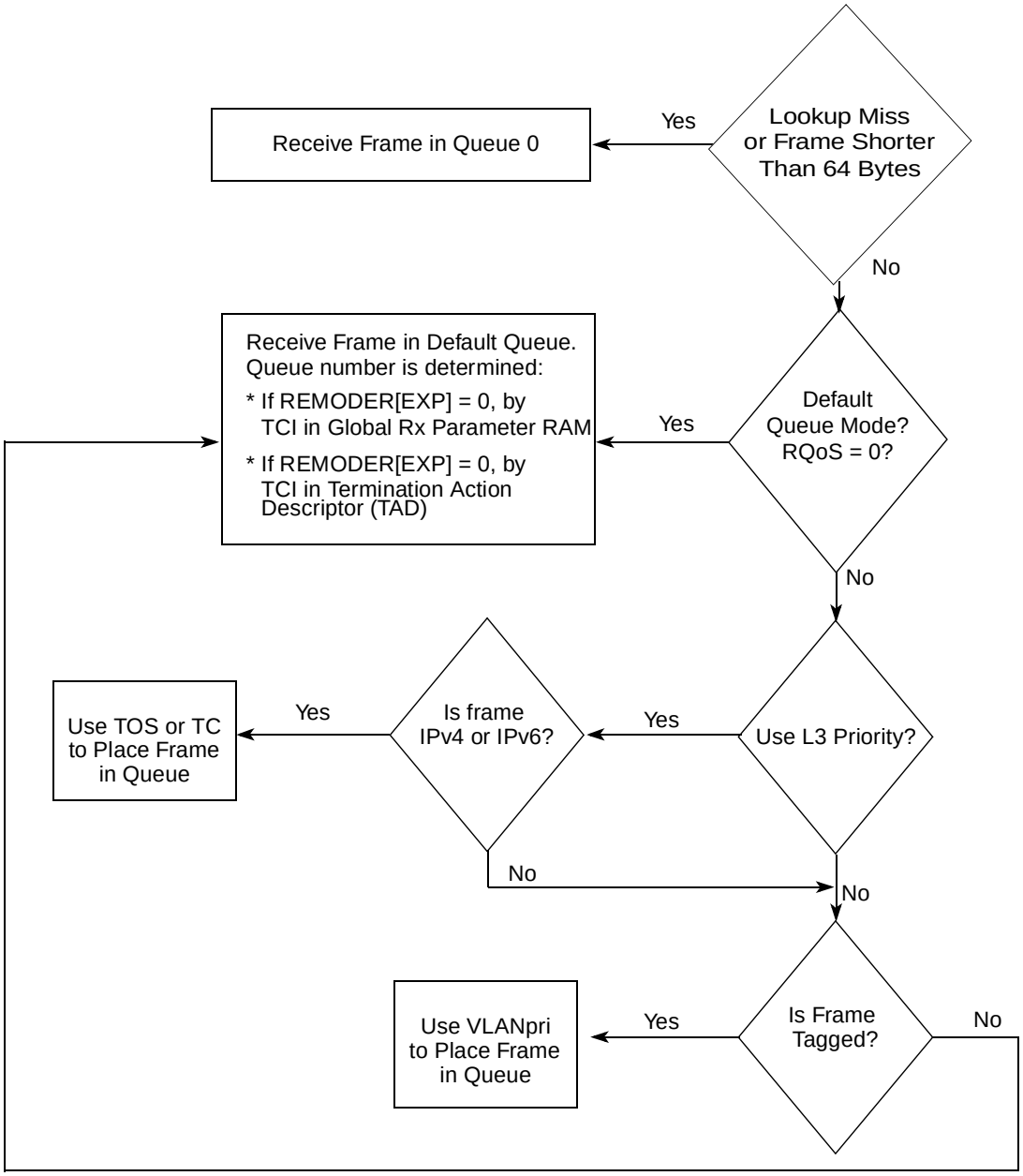


Figure 30-9. Receive Queueing Decision

30.4.13 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.

30.4.14 Align IP address

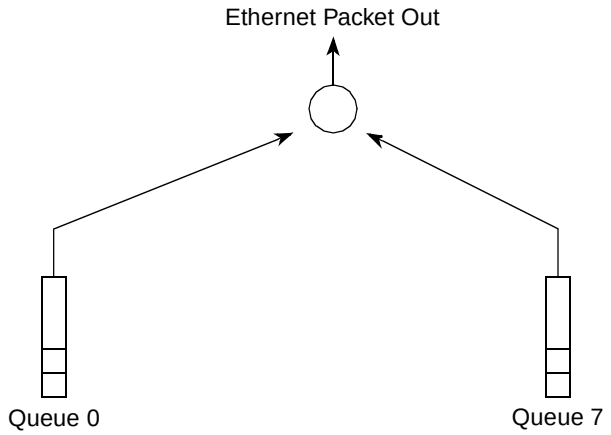
The UCC Ethernet Controller (UEC) supports IP address alignment to four bytes on received frames. If enabled, the UEC places the first byte of the frame two bytes from the address pointed by the pointer in the RxBD. The first two bytes in the data buffer contain undefined data.

30.4.15 Statistics Gathering

The UCC Ethernet Controller (UEC) implements a set of statistical counters. These counters allow the implementation of MIBs to support IEEE Std. 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.

30.4.16 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 determines the next packet to be transmitted on the shared transmitter.

Figure 30-10. Ethernet Scheduler Role

30.4.16.1 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 1Gbps 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 description of CETSCR2. See Section 20.3.9, “QUICC Engine Time-Stamp Control Register (CETSCR).”

30.4.16.1.1 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:

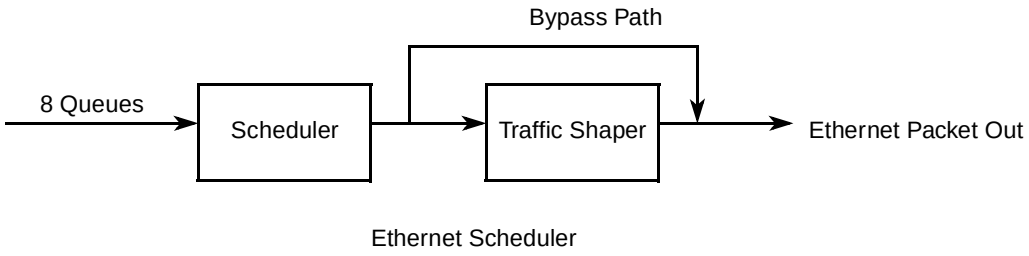


Figure 30-11. Ethernet Scheduling System

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.

30.4.16.1.2 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:

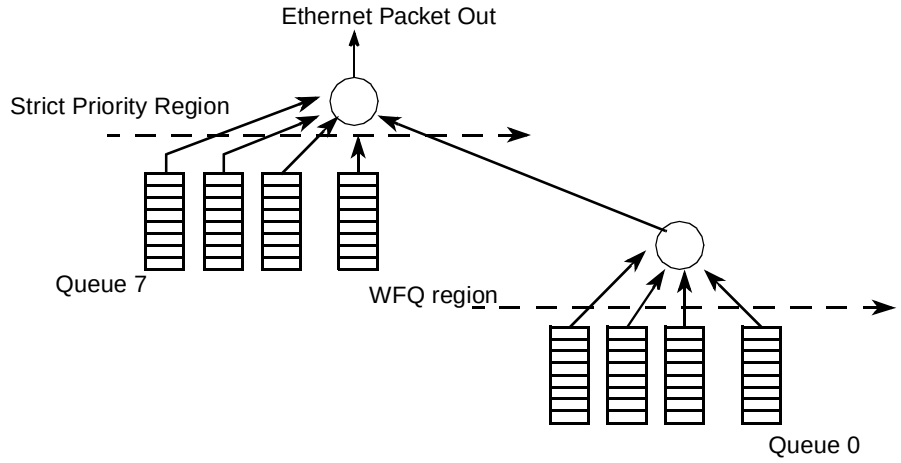


Figure 30-12. Sample Scheduler Configuration

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.

30.4.16.1.3 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.

30.4.16.1.4 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 Std. 802.3-2202 between them) in order to increase line efficiency.

Thus, the Traffic Shaper supports Token Bucket shaping with peak rate of 1Gbps, and programmable max burst length and programmable average rate.

30.4.16.1.5 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.

30.4.16.1.6 Traffic Shaper Bypass

Traffic shaper can be canceled. In this mode, the scheduling is work conserving, based on the scheduler module only. The user should expect 1Gbps bursts in any length, except from 12 bytes IPG, as defined by IEEE Std. 802.3.

30.4.16.1.7 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, i.e. 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.

System behavior will be as follows: If the Traffic Shaper is programmed to 0.5Gbps, and the traffic is queue 7 is 0.1Gbps, the total BW of the line will be 0.5Gbps+ Queue 7 rate that is 0.6Gbps. However, the traffic in Queue 6 + queues 5-0 will be limited to 0.5Gbps by the Traffic Shaper.

Queue 6 will transmit instantly, since its in the Transmit ASAP mode; However, its BW will be deduce from the total BW of 0.5Gbps.

Note, however, that if the total BW of the “Transmit ASAP” queues is higher than the total BW budget, than “Transmit ASAP” has a priority (BW is not kept).

See Section 30.15, “Traffic Shaper Programming Considerations,” for example of scheduler programming.

30.4.16.1.8 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.

30.4.16.1.9 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 30.5.3.3.3, “Scheduler Programming Model and Data Structures” for the scheduler programming model.

30.4.17 IEEE Standard 1588 Support

The UEC provides the hooks for the support of IEEE Std. 1588.

The 1588 HW assistance is supported on the following interfaces:

- MII 10/100Mbps (full and half duplex)
- RMII 10/100Mbps (full and half duplex)
- GMII/RGMII 1000Mbps (full duplex)
- RGMII 10/100Mbps (half and full duplex)

The following describes the main actions performed for each operation mode when a PTP frame was received. Frame Reception

30.4.17.1 In_band Mode

This mode is determined in the TSMR (see 1588 spec) register. The receiver’s timestamps are transferred as part of the data payload. The user sets the PTP bit in the Tx BD while sending a PTP frame, and gets a PTP indication in the Rx BD when a PTP frame has received.

The receive TSU (time stamp unit) is monitoring the received Ethernet frame on the physical line. When SFD is detected the 1588 Rx TSU locks the Real Time Value of the 1588 clock and then notifies the Ethernet MAC that SFD is detected and the timestamp associated with it is ready.

The Ethernet MAC stores the 64bit timestamp in the Memory as part of the frame at the first eight bytes of each Ethernet frame regardless of PTP indication.

When the Rx TSU parse unit is detecting a PTP frame it will notify the Ethernet MAC that a PTP frame has detected and the Ethernet controller will set RxBD[RPTP]

When the Software detects that a PTP frame has arrived by getting an interrupt or by polling the RxBD[RPTP] it will search for the PTP frame type and read the timestamp located in the first eight bytes of each frame in the memory if needed.

In case the PTP frame is marked with CRC, Overrun or Non-octet receive errors the interrupt in the TMR_PEVNT will not be set and the software should drop this frame regardless of the indication in the RxBD[RPTP]

Note that in order to work in in_band mode the following bits should be set:

UPSMR[PTPE] - 1588 enable

TSMR[opmode] - inband mode

MACCFG2[SRP] - soft receive preamble

When working in 1588 inband mode the customize preamble mode is not supported

30.4.17.2 Out_band Mode

The receive TSU (time stamp unit) is monitoring the received Ethernet frame on the physical line. When SFD is detected the 1588 Rx TSU locks the Real Time Value of the 1588 clock.

If the Rx TSU parse unit is detected a PTP frame it will notify the Ethernet MAC that a PTP frame has detected and the Ethernet controller will set RxBD[RPTP]. In addition, the Rx TSU will set the corresponding bit in the event register and generate an interrupt to the CPU.

When the Software detects that a PTP frame has arrived by getting an interrupt or by polling the RxBD[RPTP] it will read the dedicated timestamp register(TMR_UCx_RXTS_H/L) through the host interface.

In case the PTP frame is marked with CRC, Overrun or Non-octet receive errors the interrupt in the TMR_PEVNT will not be set and the software should drop this frame regardless of the indication in the RxBD[RPTP]

30.4.17.3 PTP frame transmission

The following describes the main actions performed for each operation mode when a PTP frame is transmitted:

30.4.17.3.1 In_band Mode

If the Software transmits a PTP frame, it should set the appropriate bit in the buffer descriptor TxBD[TPTP]. The Ethernet MAC is responsible to indicate the Tx TSU that a PTP frame is transmitted.

The transmit TSU (time stamp unit) is monitoring the transmitted Ethernet frame on the physical line. When SFD is detected the 1588 Tx TSU locks the Real Time Value of the 1588 clock. If the Tx TSU gets a PTP frame indication it sets the corresponding bit in the event register and generates interrupt to the CPU.

The SW should access the dedicated timestamp register (TMR_UCx_TXTS_H/L) through the host interface after an interrupt generation or when the last bit in the TxBD is set (TxBD[last])

30.4.17.3.2 Out_band Mode

Same as inband mode

30.4.18 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

30.4.19 UEC Physical Interfaces

The physical interface is user programmable by configuring the PSMR, MACCFG2 registers.

30.4.19.1 10 and 100 Mbps MII Interface Operations

The MII is the media independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the 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.

30.4.19.2 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)

30.4.19.3 1000 Mbps GMII and TBI Interface Operations

The GMII is the gigabit media independent interface defined by the 802.3 standard for 1000 Mbps operation. Independently, the MAC-PHY interface can also operate in TBI mode. Note that either the TBI or GMII interface is chosen and not both at the same time. TBI is the ten-bit interface which contains PCS functions (10-bit encoding/decoding) as defined by the 802.3z standard. The UCC Ethernet Controller provides the GTX_CLK to the PHY in GMII or TBI mode of operation.

30.4.19.4 1000 Mbps RGMII and RTBI Interface Operations

The RGMII and RTBI are intended to be an alternative to the GMII and TBI interfaces respectively. The principle objective is to reduce the number of pins required to interconnect the MAC and PHY. In order to accomplish this objective, the data paths and all associated control signals are reduced and control signals are multiplexed together and both edges of the clock are used. For gigabit operation, the clocks operate at 125 MHz.

30.4.20 Diagnostic Modes

30.4.20.1 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 GMII,MII,RGMII is connected internally to the receiver input, and RXD is ignored.

In TBI mode internal loopback mode is selected by programming the LoopBack bit in CR register (See Section 30.11.7.4, “Control Register (CR)”). In RTBI mode loopback is not supported.

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.

30.4.20.2 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

30.4.20.3 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 RGMII 10/100 Mbps and 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 (10/100/1000 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. Note that the GMII, TBI and RTBI interfaces support only the full duplex mode.

30.5 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.

30.5.1 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 the IEEE 802.3 standard. All of the 32 bit MAC registers must be accessed as 32 bit entities.

30.5.1.1 UCC Protocol Specific Ethernet Mode Register (UPSMR)

In Ethernet mode, the UCC protocol-specific mode register, shown in Figure 30-13, functions as the Ethernet mode register.

NOTE

This register must be initialized after the GUEMR Register, see Section 23.3.2, "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	FTFE	PTPE	Reserved			ECM	HSE	0	Res	PRO	—	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		BRO									
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 30-13. UCC Ethernet Mode Register

Table 30-5. UPSMR Ethernet Field Descriptions

Bits	Name	Description
0	FTFE	Flush Tx FIFO when a Flow Control frame is received. This bit must be set when running the L2 switch. Otherwise it should be cleared. If this bit is set and a Flow Control frame is received, the UEC send the current frame till completion, and then flushes the frames in the transmit FIFO. No transmission occurs till the MAC Parameter time has elapsed.
1	PTPE	PTP Enable 0 - Disable IEEE Std. 1588 assist connection to MAC I/F 1 - Enable IEEE Std. 1588 assist connection to MAC I/F
2-4	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 30.7.5, "UCC Statistics (Hardware Counters)." 1 - Perform hardware statistics. This bit is equivalent to the RMON bit in MPC82xx.

Table 30-5. UPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
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.
11	RSH	Receive short frames 0 Discard short frames (frames smaller than the value specified in MINFLR) 1 Receive short frames
12	RPM	GMII/TBI Reduced pin mode interfaces. See Table 30-6 for usage of this bit.
13	R10M	RGMII/RMII 10/100 mode. See Table 30-6 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	Ten-bit interface mode. See Table 30-6 for usage of this bit.
16–17	AUFC	Automatic Flow Control Enable 00 - No Automatic Flow Control frames are sent by the UEC. CPU may initiate a transmit flow control frame issuing START FLOW CONTROL command. 01 - In full duplex mode, the UEC sends a pause frame when RxFIFO reaches its special emergency threshold (that is, URFSET is crossed upwards). When the RxFIFO empties out below the emergency threshold (that is, URFET is crossed downwards), the UEC automatically sends pause frame with MAC parameter duration of “zero pause quantas”. It is assumed the URFET < URFSET. In half-duplex mode, when URFSET is crossed, the MAC applies back pressure algorithm on the line by sending preambles on the line when no data is available for transmit; when URFET is crossed downwards, the UEC disables the back pressure algorithm. See Section 27.5.4, “Receive Virtual FIFO Special Emergency Threshold (URFSET). 10 - Reserved (not allowed) 11 - In addition to the behaviour described when AUFC = 01, the UEC sends 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.
18	Res (1)	Must be set to ONE.
19	RMM	RMII Mode. See Table 30-6 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 30-14

Table 30-5. UPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
21	Reserved	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

The following Table summarizes the register programming needed to configure the UEC in the desired mode. Only the combinations in this table are allowed:

Table 30-6. Interface Mode Configuration 10/100

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

Table 30-7. Interface Mode Configuration GEnet

Mode	MACCFG2[IFM]	UPSMR[RPM]	UPSMR[TBIM]	UPSMR[R10M]	UPSMR[RMM]	MACCFG2[FDX]
GMII	10	0	0	0	0	1
TBI	10	0	1	0	0	1
RTBI	10	1	1	0	0	1
RGMII(1000)	10	1	0	0	0	1
RGMII (100)	01	1	0	0	0	1
RGMII(10)	01	1	0	1	0	1

The behavior of the UCC Ethernet Controller is not specified if programmed to other combinations of these bits.

30.5.1.2 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 Ethernet mask register (UCCM).

The UCCM has the same bit format as UCCE, refer to Figure 30-14. Setting an 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 30-14. Ethernet Event/Mask Register (UCCE/UCCM)

Table 30-8 describes the UCCE/UCCM fields.

Table 30-8. 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 (i.e.the receiver has signalled to the transmitter to enter backpressure. See Section 27.5.4, “Receive Virtual FIFO Special Emergency Threshold (URFSET)” for FIFO thresholds that triggers this event); OR the UEC has exited the backpressure state (i.e. the RxFIFO is filled below the threshold programmed in the URFET register, see Section 27.5.3, “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 (i.e. it has received a PAUSE frame and its transmitter is currently not transmitting on the line); OR the UEC has exited a PAUSE state (i.e. 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

Table 30-8. UCCE/UCCM Mode Register Descriptions (continued)

Bits	Name	Description
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 respictive RxBD[I] bit is set. If REMODER[IWEn]=1 these bits are used as follows:
24:31	RXF0-7	Rx frame. Set when a complete frame is received on the Ethernet channel on queue number 0-7 respectively.

30.5.1.3 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 30-15. Ethernet Status Register (UCCS)

Table 30-9 describes the UCCS fields.

Table 30-9. 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

30.5.1.4 MAC Configuration 1 Register (MACCFG1)

MAC configuration registers (MACCFG1 and 2) provide the means of configuring the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal 7 bytes to some other (non-zero) value.
- Varying pad/CRC combinations—Three different pad/CRC combinations are provided to handle a variety of system requirements. Most simple are frames that already have a valid FCS field. In this case, CRC is checked and reported via 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 30-16 describes the definition for the MACCFG1 register.

Offset	UCC EMAC1:UCCx_base + 0x100; UCC EMAC2:UCCx_base + 0x200															
	0	1	2								11	12	13	14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16						22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	Rx_Flow	Tx_Flow	Sync'd Rx EN	Rx_EN	Sync'd Tx EN	Tx_EN
W																
Reset	0000_0000_0000_0000															

Figure 30-16. MACCFG1 Register Definition

Table 30-10 describes the MACCFG1 fields.

Table 30-10. 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.

Table 30-10. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
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.

30.5.1.5 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. Figure 30-17 describes the definition for the MACCFG2 register.

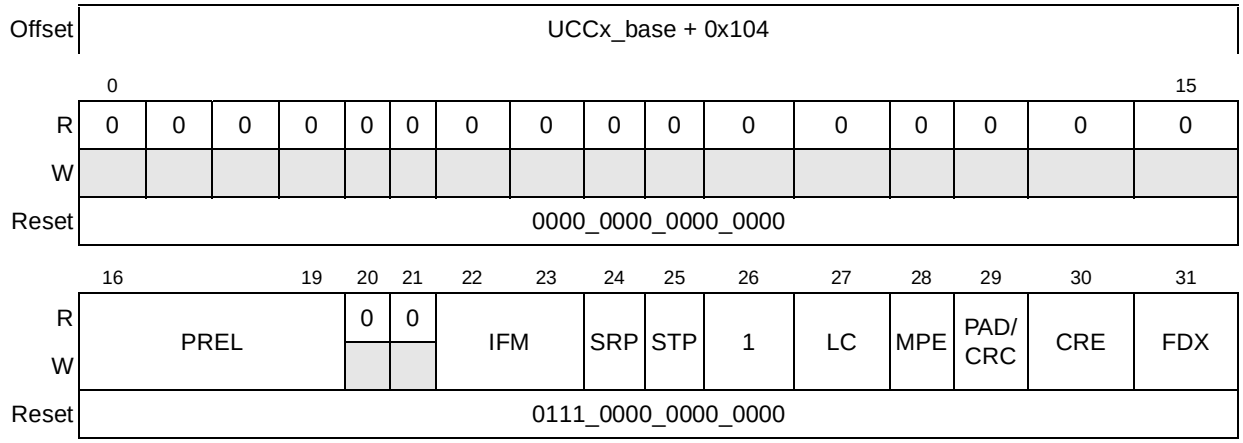


Figure 30-17. MACCFG2 Register Definition

Table 30-11 describes the MACCFG2 fields.

Table 30-11. 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

Table 30-11. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
22–23	IFM	Interface Mode. This field determines the type of interface to which the MAC is connected. Its default is 00. See Table 30-6 for usage of this bit. 00 Reserved 01 Nibble mode (MII/RMII/RGMII 10/100bps) 10 Byte mode (GMII/TBI/RTBI/RGMII 1000bps) 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 TBI or RMII this bit should be cleared
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 TBI or 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.

Table 30-11. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
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.

30.5.1.6 Inter-frame Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user.

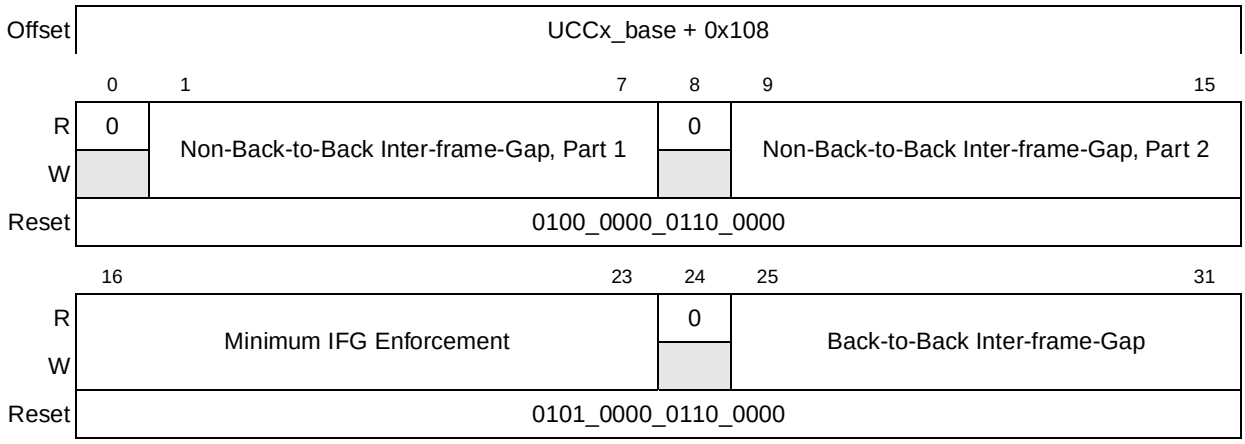


Figure 30-18. IPGIFG Register Definition

Table 30-12 describes the IPGIFG fields.

Table 30-12. 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 Std. 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[1]	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. Zero is not a legal value.
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.

[1] Value zero is restricted from use.

30.5.1.7 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 30-19 describes the definition for the HAFDUP register.

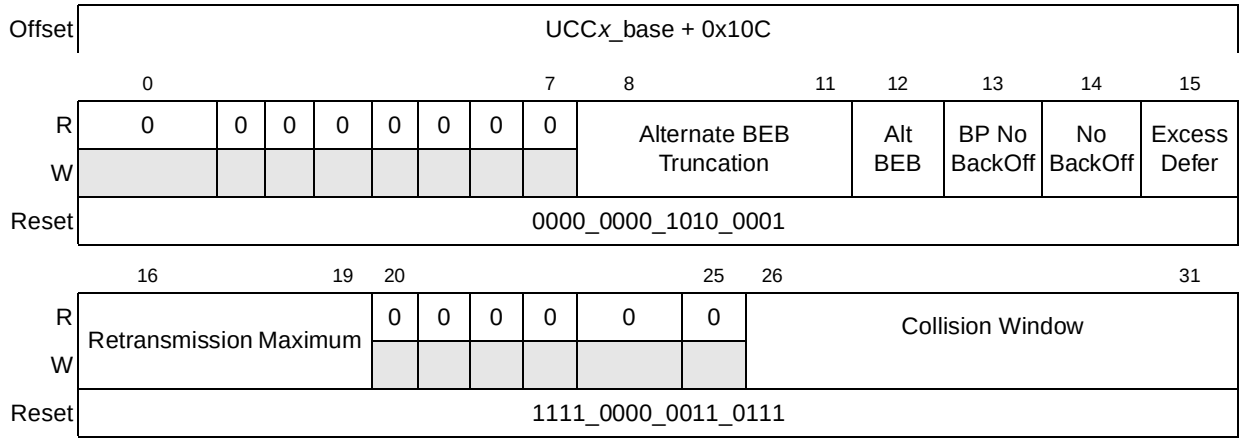


Figure 30-19. Half-Duplex Register Definition

Table 30-13 describes the HAFDUP fields.

Table 30-13. 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 30-13. 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.

30.5.1.8 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 30-20 describes the definition for the MIIMCFG register.

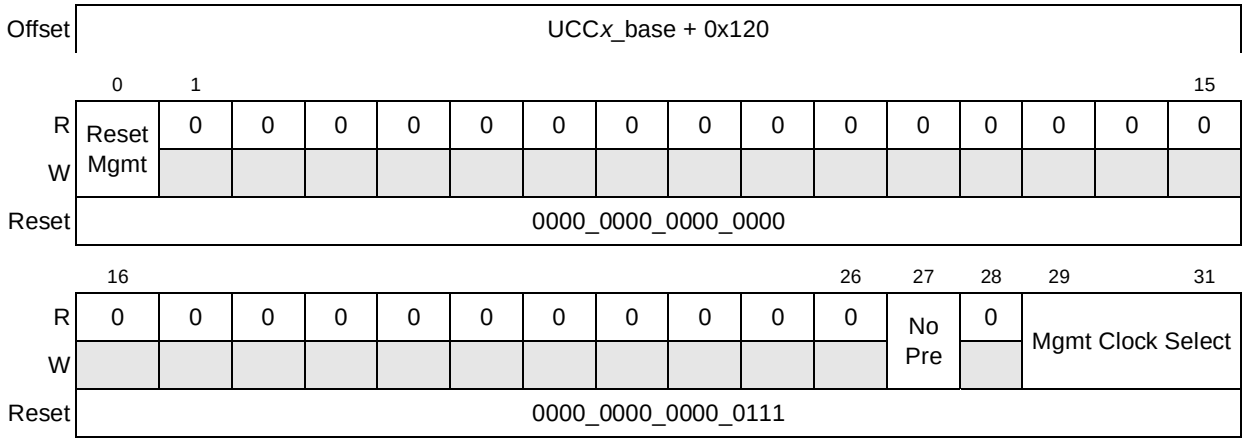


Figure 30-20. MII Management Configuration Register Definition

Table 30-14 describes the MIIMCFG fields.

Table 30-14. 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 Std. 802.3/22.2.4.4.2.

Table 30-14. 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

30.5.1.9 MII Management Command (MIIMCOM)

The MIIMCOM register is written by the user. Figure 30-21 describes the definition for the MIIMCOM register.

Offset: UCC*x*_base + 0x124

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Scan Cycle	Read Cycle
W																
Reset	0000_0000_0000_0000															

Figure 30-21. MIIMCOM Register Definition

Table 30-15 describes the MIIMCOM register.

Table 30-15. MIIMCOM Descriptions

Bits	Name	Description
0–29	—	Reserved

Table 30-15. MIIMCOM Descriptions (continued)

Bits	Name	Description
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].

30.5.1.10 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 30-22 describes the definition for the MIIMADD register.

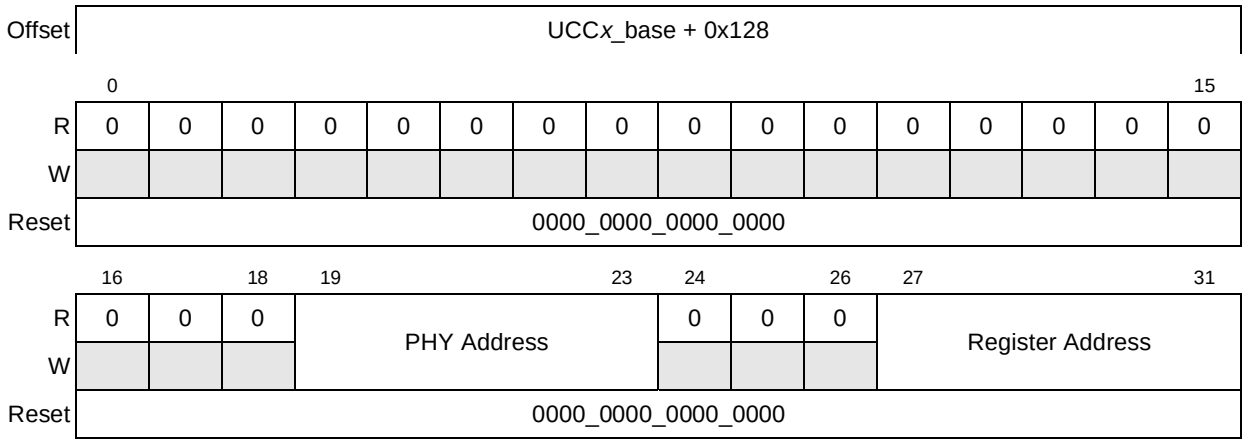


Figure 30-22. MIIMADD Register Definition

Table 30-16 describes the MIIMADD fields.

Table 30-16. 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). Reprogramming 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.

30.5.1.11 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 30-23 describes the definition for the MIIMCON register.

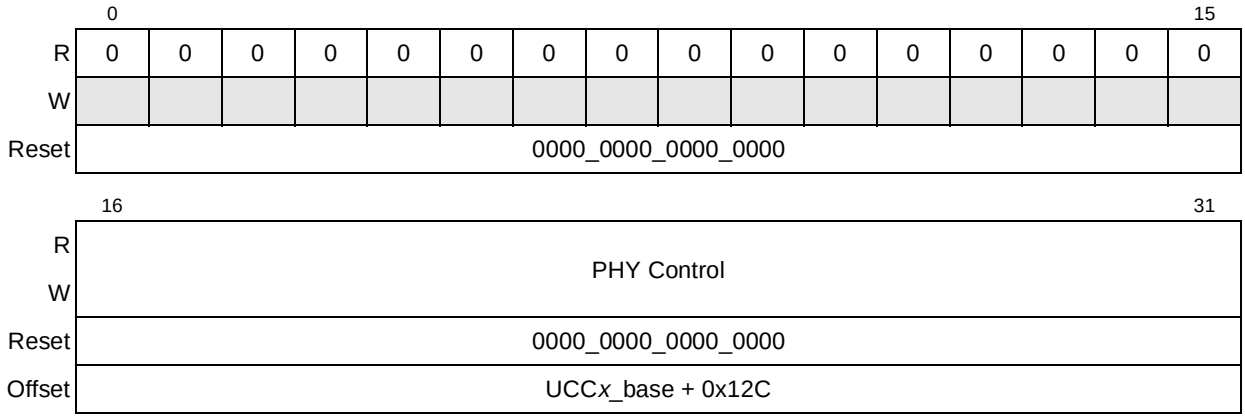


Figure 30-23. MII Mgmt Control Register Definition

Table 30-17 describes the MIICON fields.

Table 30-17. 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.

30.5.1.12 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read-only by the user. Figure 30-24 describes the definition for the MIIMSTAT register.

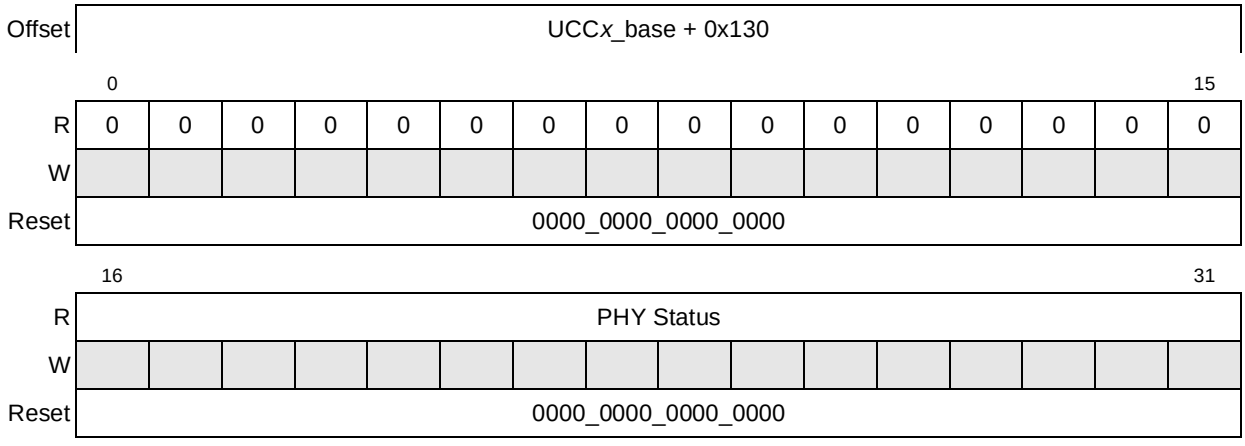


Figure 30-24. MIIMSTAT Register Definition

Table 30-18 describes the MIIMSTAT fields.

Table 30-18. 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.

30.5.1.13 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 30-25 describes the definition for the MIIMIND register.

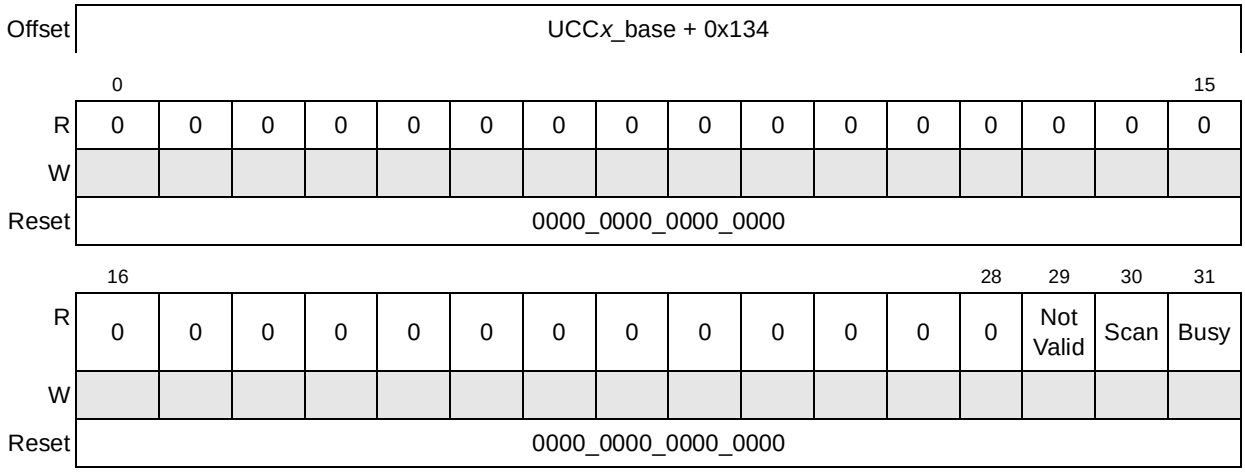


Figure 30-25. MII Mgmt Indicator Register Definition

Table 30-19 describes the MIIMIND fields.

Table 30-19. 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.

30.5.1.14 Interface Status Register (IFSTAT)

The IFSTAT register is read by the user. Figure 30-26 describes the definition for the IFSTAT register.

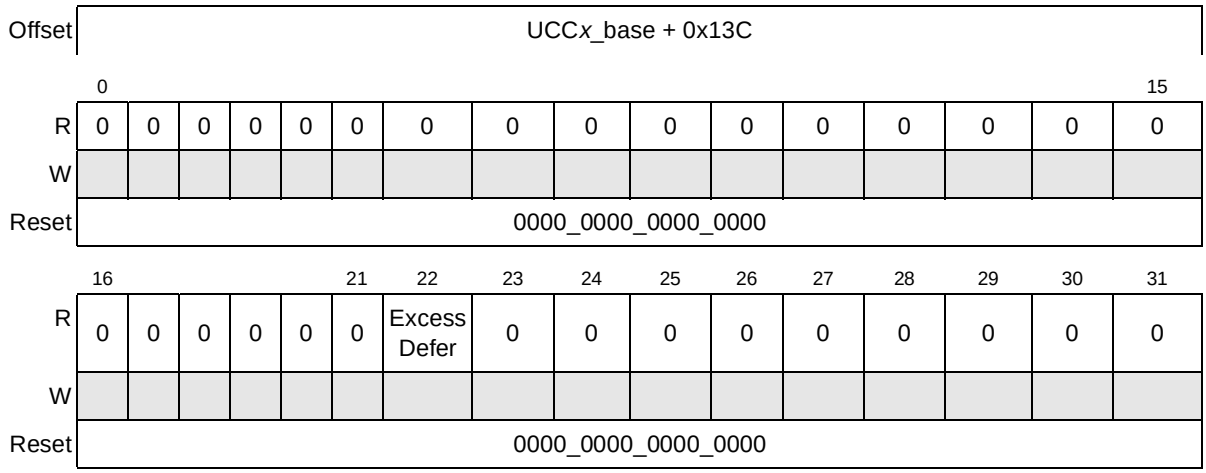


Figure 30-26. Interface Status Register Definition

Table 30-20 describes the IFSTAT fields.

Table 30-20. 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

30.5.1.15 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 30-27 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.

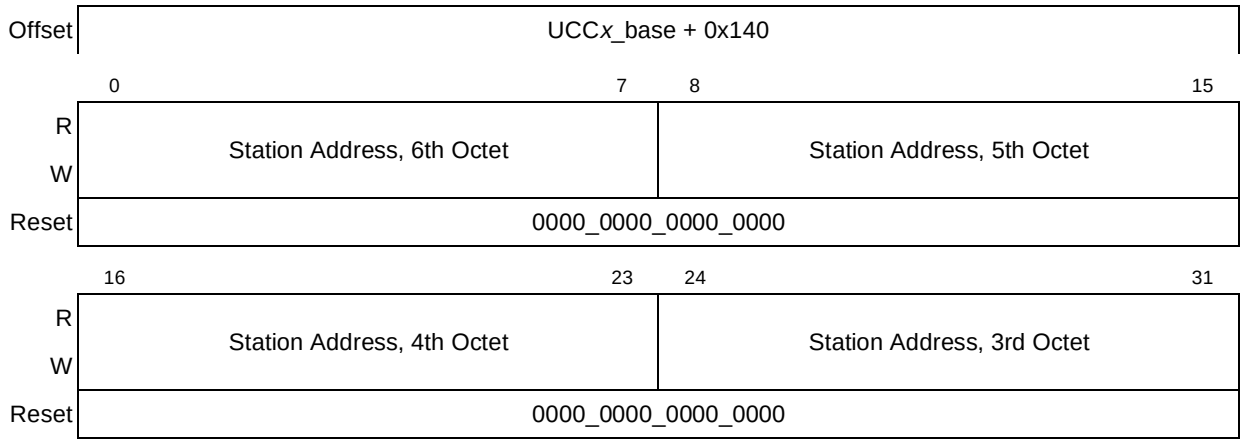


Figure 30-27. Station Address Part 1 Register Definition

Table 30-21 describes the MACSTNADR1 fields.

Table 30-21. 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.

30.5.1.16 Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 30-28 describes the definition for 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.

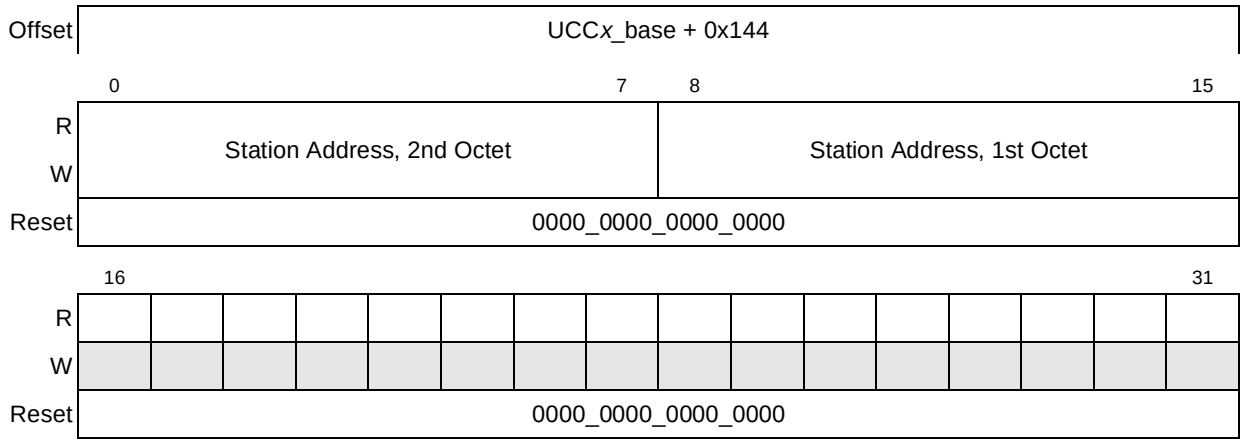


Figure 30-28. Station Address Part 2 Register Definition

Table 30-22 describes the MACSTNADDR2 fields.

Table 30-22. 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

30.5.1.17 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 30-29. UCC Ethernet Mac Parameter Register (UEMPR)

Table 30-23. 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. Can contain any user-defined data. Note that this does not extend the MAC parameter value, which would cause more delay when a flow control frame is detected.

30.5.1.18 UCC Ten Bit Interface Physical Address Register (UTBIPAR)

offset	UCC*x*_base + 0x154															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved															
R/W	R															
Reset	0000_0000_0000_0000															
Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	Reserved											**TBIPA**				
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-30. UCC Ten Bit Interface Physical Address Register (UTBIPAR)

Table 30-24. UTBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27-31	TBIPA	This field is used to program the PHY address of the ten-bit interface s MII management bus. To access the TBI register the user must write the TBIPA value to the MIIMADD [PHY Address] register located in the MAC register, MII Management Address Register (MIIMADD).

30.5.1.19 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 30-31. Ethernet Event/Mask Register (UESCR)

Table 30-25 describes the UESCR fields.

Table 30-25. 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 QE. The user should not alter the value of this field. 0x0 value is not legal.

30.5.2 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.

30.5.2.1 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 (i.e. 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 30-32 TxBD[VID] field.
- The BD ring size in units of BDs must be greater than the maximum number of TxBDs that construct a frame (i.e., 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 (i.e. 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 30-32. 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 / PTP	**IPCHI0 /** RL	RC/VID				**IPCHI1** / UN	**IPCHI2 /** CSL
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

Table 30-26. Transmit Buffer Descriptor Field Description

Offset	Bits	Name	Description
Offset + 0	0	R	Ready, written by UCC Ethernet Controller and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The 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.
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.
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.

Table 30-26. Transmit Buffer Descriptor Field Description (continued)

Offset	Bits	Name	Description
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. PTP (written by CPU) 0 No PTP action 1 This frame is a IEEE Std. 1588 PTP frame. This frame is timestamped by the UEC at the time it transmitted on the line, and if unmasked, an interrupt is issued to the CPU. This bit is set only on the first TxBD of a frame. See Chapter 31, "QUICC Engine IEEE1588 Assist."
Offset + 0	9	IPCH0/ RL	IPCHI0, IPCHI1 (bit 14), IPCHI2 (bit15). Written by the CPU. Number of entry in IPH_Offset in Section 30.5.3.3, "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.

Table 30-26. Transmit Buffer Descriptor Field Description (continued)

Offset	Bits	Name	Description
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 VTBL_BASE. 0xxx - Do not Insert a VLAN Tag on the Transmitted Frame 1nnn- Use VLAN Tag at index VTBL_BASE+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.
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.

30.5.2.2 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 (i.e. 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
- MRBLR must be a multiple of Virtual FIFO block size (i.e. a multiple of 128 bytes)

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

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 30-33 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	PTP	M	BC	MC	LG	NO	SH	CR	OV	IPCH
Offset + 2	DATA LENGTH															
Offset + 4	RX DATA BUFFER POINTER															
Offset + 6																

Figure 30-33. Receive Buffer Descriptor

Table 30-27. 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]).

Table 30-27. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
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	PTP	If IEEE Std. 1588 mode is enabled in the UCC this bit is set by the UCE if this is a PTP frame. Refer to Chapter 31, "QUICC Engine IEEE1588 Assist." In this case LAST_PCD[BDM0] feature is not supported. If IEEE Std. 1588 Mode is disabled in the UCC this bit is set as follows. If REMODER[EXP]=1. Valid only when L bit is set in the RxBD This bit is set if LAST_PCD[BDM0] is set
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).
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.

Table 30-27. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	15	IPCH	IP Checksum written by the UCC Ethernet Controller if REMODER[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 REMODER[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 8-byte aligned. The buffer must reside in memory external to the UCC Ethernet Controller.

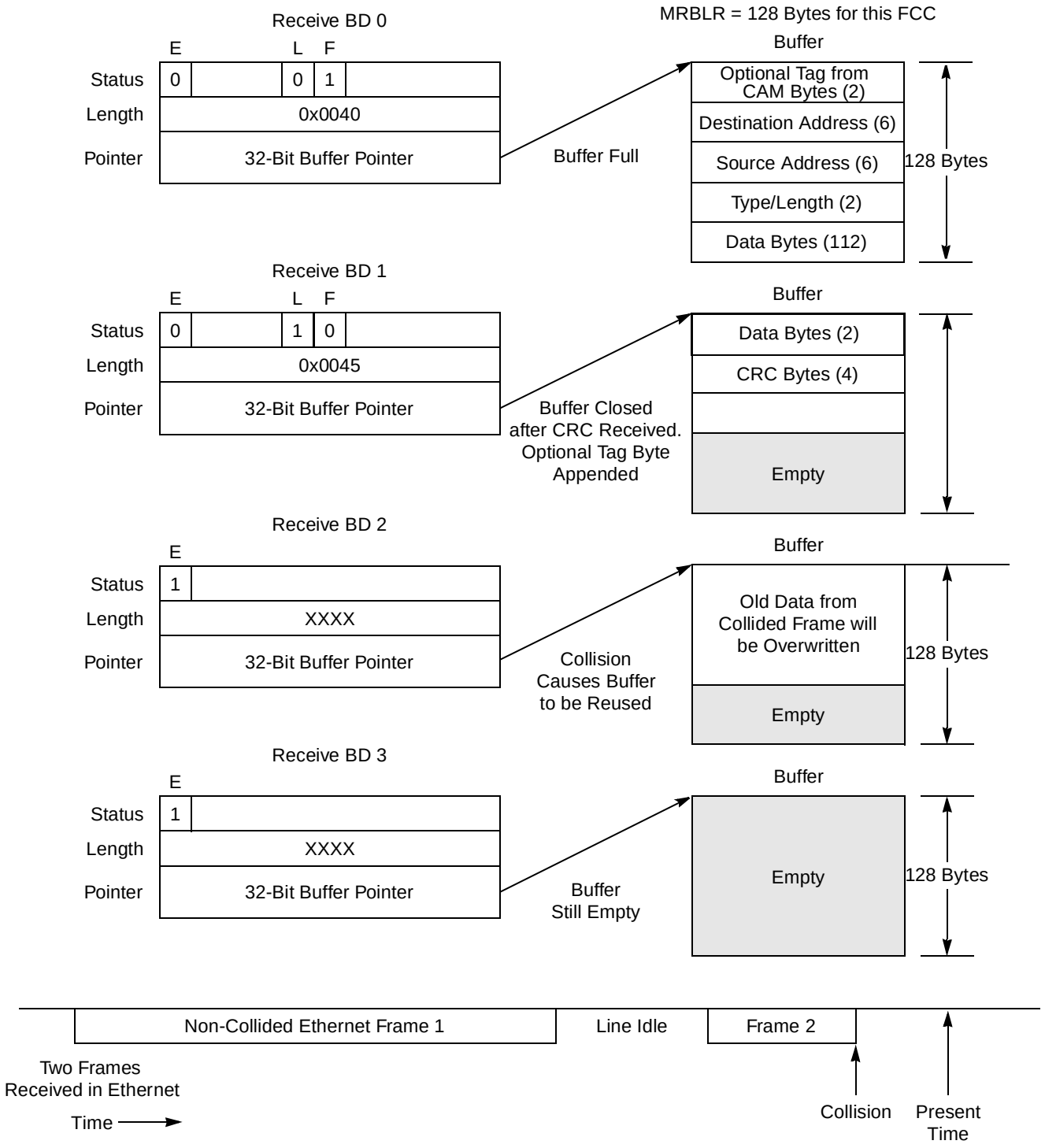


Figure 30-34. Example of Ethernet Receiving Using RxBD

30.5.3 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 QE Commands “Section 30.8.1, Init Tx, Init Rx and InitTx and Rx Parameters Command” on page 115. 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.

The following table specifies the multiuser RAM allocation for data structures necessary for the UEC as a function of the number of queues and the nominal bit rate on the interface. One data structure is associated with the nominal bit rate which is programmed in INIT RX PARAMETERS[RGF] and INIT TX PARAMETERS[TGF] fields. The other data structure is associated with the number of queues, which is programmed in REMODER[NumOfQueues] and TEMODER[NumOfQueues] fields.

Table 30-28. Rx Global Parameter RAM Allocation

	Extended Filtering	Bit Rate					Number of queues							
	REMODER [EXP]=1	1Gbps	10/100 Mbps	Other values for system optimizations			1		2		4		8	
Number of Threads		4	1	2	6	8								
Pointer Name	EXFGlobal Param	RQPTR	RQPTR	RQPTR	RQPTR	RQPTR	RBDQPTR	IntCoalescingPTR	RBDQPTR	IntCoalescingPTR	RBDQPTR	IntCoalescingPTR	RBDQPTR	IntCoalescingPTR
Global Parameter RAM size	16bytes	160 bytes	40 bytes	80 bytes	240 bytes	320 bytes	48 bytes	8 bytes	96 bytes	16 bytes	192 bytes	32 bytes	384bytes	64 bytes

The following table summarizes the memory size for the thread parameter RAM, depending on the features selected by the user. Each thread requires a separate parameter RAM of the size mentioned in this table.

Table 30-29. Rx Thread Parameter RAM allocation

Mode	Basic Thread Parameter RAM	REMODER[EXP]=1 Extended Parsing Mode			Total
	Always Needed	Lookup Type			
Termination	128[1]	No External[2] HashLookup		64	192
		ExternalHash Lookup[5]	8 byte LookupKey	128	256
			16 byte LookupKey	160	288

1 See Section 30.5.3.6, “Rx Thread Parameter RAM”

2 Optionally part of Rx Thread Parameter RAM.

The following table summarized the memory size needed for the Tx Parameter RAM.

Table 30-30. Tx Parameter RAM Allocation

	Bit Rate					Number of queues			
	1Gbps	10/100Mbps	Other values for system optimizations			1	2	4	8
Number of Threads	4	1	2	6	8				
Pointer Name	TQPTR	TQPTR	TQPTR	TQPTR	TQPTR	SQPTR	SQPTR	SQPTR	SQPTR
Multiuser RAM size	416 bytes	136 bytes	208 bytes	624 bytes	832 bytes	64 bytes	128 bytes	256 bytes	512 bytes

30.5.3.1 Transmitter Parameter RAM Overview

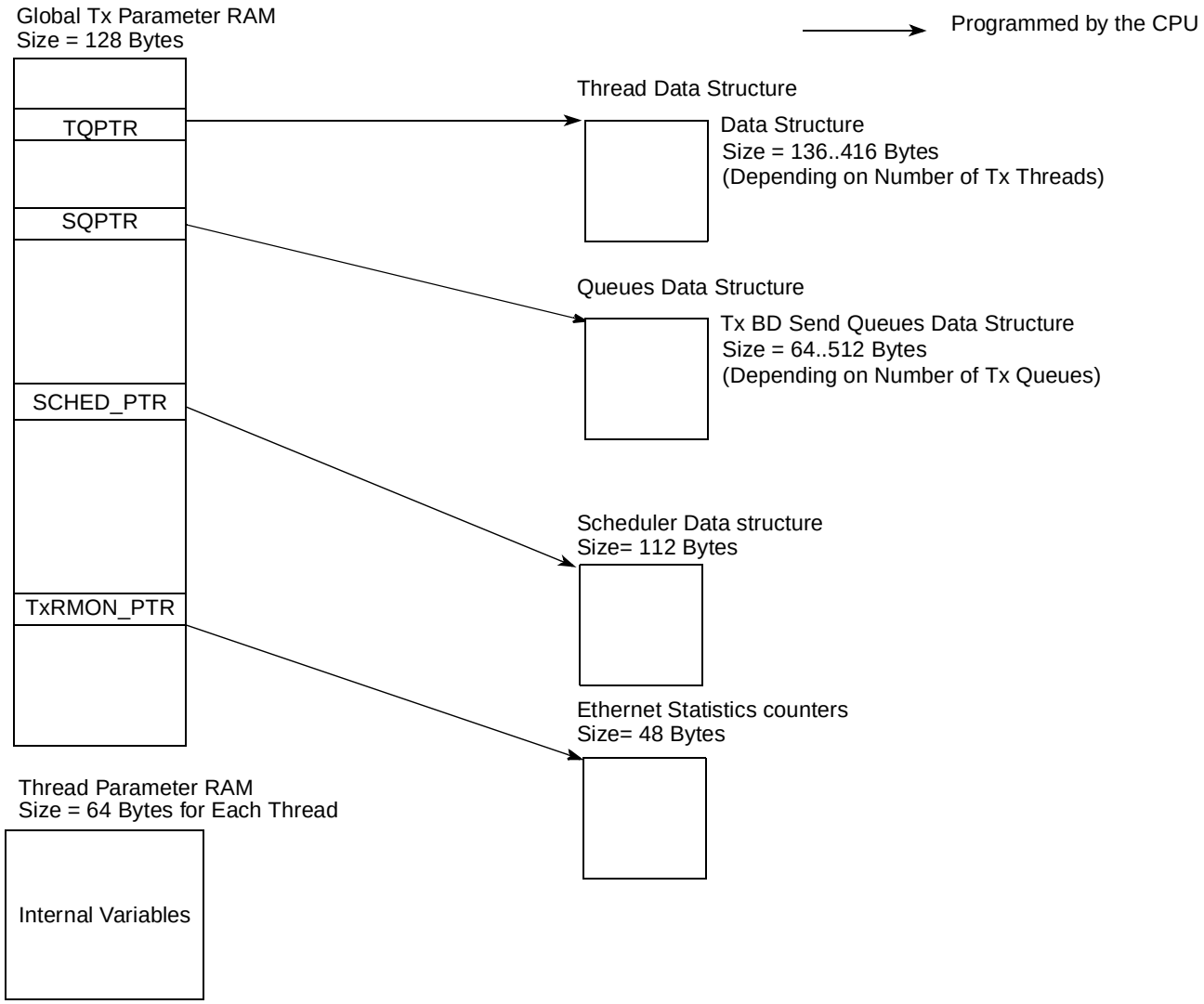


Figure 30-35. Transmitter Parameter RAM Data Structures

30.5.3.2 Receiver Parameter RAM Overview

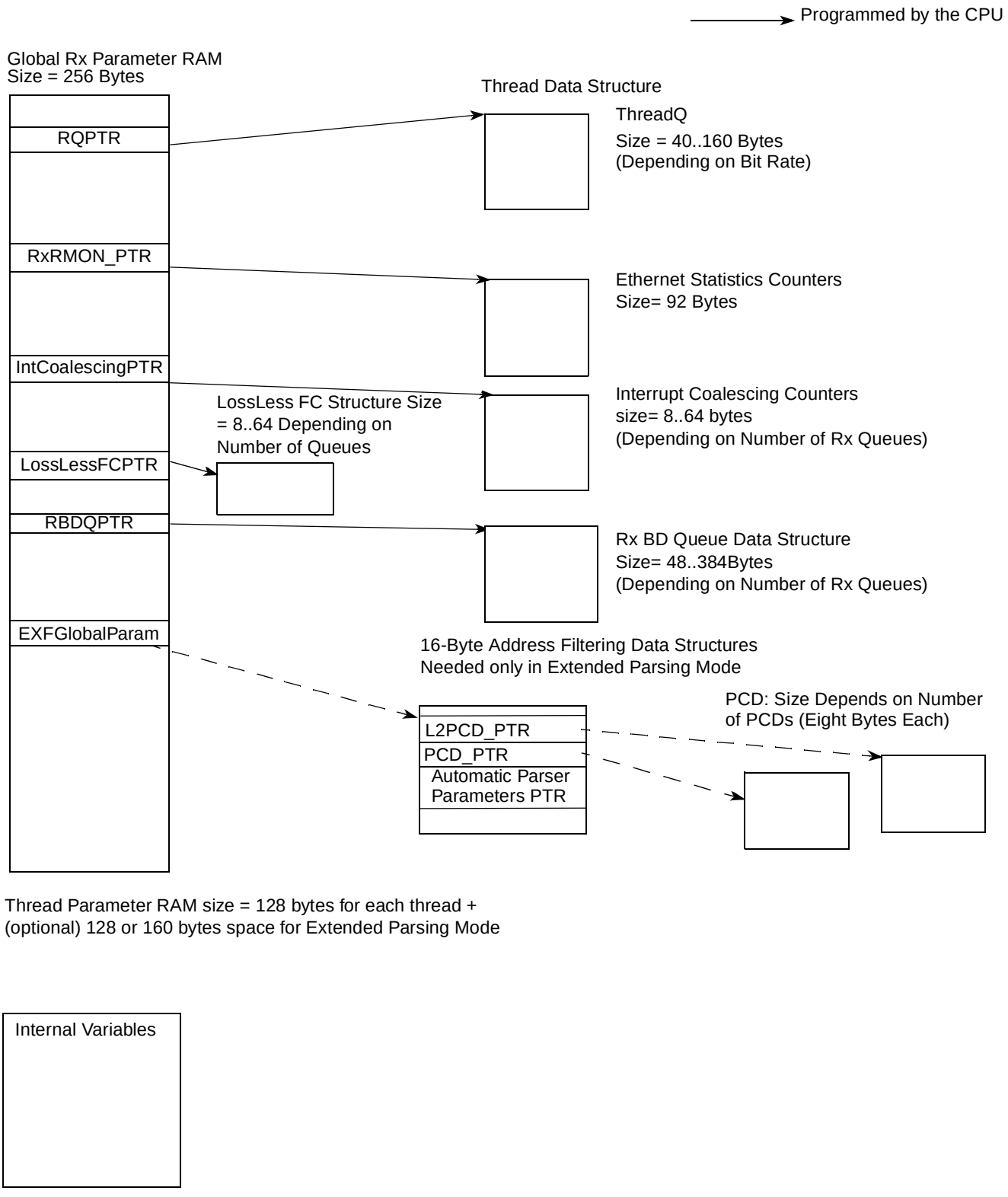


Figure 30-36. Receiver Parameter RAM Data Structures for Termination

30.5.3.3 Tx Global Parameter RAM

The following table describes the Tx Global Parameter RAM:

Table 30-31. 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	53 bytes	Reserved		QE
0x38-0x3B	0:15	SQPTR	Base address of the send queue memory region. See Section 30.5.3.3.2, “Tx Send Queue Memory Region.”	CPU
	0:15			
0x3C-0x3F	0:15	SchedulerBasePointer	Base address of the scheduler memory region (SCBASE) located in internal Multiuser RAM. See Section 30.5.3.3.3, “Scheduler Programming Model and Data Structures”. This address must be eight bytes aligned.	CPU
	0:15			
0x40-0x43	0:15	TxRMONBasePointer	Base address of the Tx RMON statistic counter located in internal Multiuser RAM. See Section 30.7.4.1, “TX Firmware Counters.” This address must be four bytes aligned.	CPU
	0:15			

Table 30-31. 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 QE usage. See Section 30.5.4, "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	0:31	TQPTR	Base Address of theThread Data Structure located in internal Multiuser RAM. 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

30.5.3.3.1 Thread Data Structure

This data structure is located at base address TQPTR. The data structure is initiated by the QE upon issuing an INIT Tx Ethernet Command.

30.5.3.3.2 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 the following table:

Table 30-32. 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 the following table:

Table 30-33. Tx Send Queue Descriptor (SQQD)

Offset	Bits	Name	Description (Data structure must be 32 bytes aligned).	Initialized by
0x00-0x03	0:31	BDRingBase	A pointer to the BD ring base address located in external memory. This field must be initialized to the base address of the BD ring.	CPU
0x04-0x07		Reserved	Set to zero	QE
0x08-0x09	0:31	BDRConsumerAddress	Points to the next entry in the BD ring to be executed. This field must be initialized after reset to the first entry in the BD ring.	QE
0x0A - 0x0B		Reserved		QE
0x0C - 0x0D	0:15	LastBDCompletedAddress	This field must be initialized after reset to point to the last entry in the BD ring.	CPU
0x0E-0x3F		Reserved	Set to zero	QE

30.5.3.3.3 Scheduler Programming Model and Data Structures

The following tables summarizes the programming model and data structures. The base address to the scheduler memory region is specified in the Tx Global parameter RAM.

Table 30-34. 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	QE Packet counter. This entry is updated by the QE. The CPU must no alter its value.	
0x6	CECount1	16	QE Packet counter. This entry is updated by the QE. 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	QE Packet counter. This entry is updated by the QE. The CPU must no alter its value.	
0xE	CECount3	16	QE Packet counter. This entry is updated by the QE. 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	QE Packet counter. This entry is updated by the QE. The CPU must no alter its value.	

Table 30-34. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x16	CECount5	16	QE Packet counter. This entry is updated by the QE. 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	QE Packet counter. This entry is updated by the QE. The CPU must no alter its value	
0x1E	CECount7	16	QE Packet counter. This entry is updated by the QE. 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 QE	N/A
0x44	Time	32	Temporary variable handled by the QE	N/A
0x48	TTL	32	Temporary variable handled by the QE	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. Also see example calculations in Section 30.15, “Traffic Shaper Programming Considerations.”	CPU
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 30.15, “Traffic Shaper Programming Considerations.”	CPU
0x53	Reserved	8	Set to zero	CPU

Table 30-34. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
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	
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 QE. Initialized to 0.	QE

Table 30-34. Scheduler Programming Model (continued)

Address (offset from SCBASE)	Name	Width (bit)	Description	Initialized by
0x64	Reserved	32	Initialized to 0.	QE
0x68-6F	Reserved		Initialized to 0	QE

30.5.3.4 Tx Thread Parameter RAM

The TX thread parameter RAM is summarized by the following table:

Table 30-35. Tx Thread Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x00-0x3F		Reserved	Internal variable.	QE

30.5.3.5 Rx Global Parameter RAM

The RX Global Parameter RAM is summarized by the following table:

Table 30-36. RX Global Parameter RAM

Offset	Bits	Name	Description	Initialized by
0x00	0–31	REMODER	Ethernet Mode Register. See Section 30.5.3.7, “Rx Ethernet Mode Register (REMODER).” Initialize to zero.	CPU
0x04	0–31	RQPTR	Base Address of the Thread Data Structure located in internal Multiuser RAM. 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 QE 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 located in internal Multiuser RAM. This base address must be four bytes aligned.	CPU

Table 30-36. RX Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
0x28-0x2F		Reserved	Set to zero	QE
0x30	0–31	IntCoalescingPTR	Base Address to Interrupt Coalescing Table located in internal Multiuser RAM. 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
0x35	0–7	Reserved		QE
0x36	0–7	RSTATE	Receive internal state. Reserved for QE use only. This byte contains the bus mode register. See Section 30.5.4, “Bus Mode Register (BMRx).”	CPU
0x37-0x43		Reserved	Set to zero	QE
0X44	0–7	OV_SkipFrame	Number of frames to be discarded (skipped) when a Rx Virtual FIFO overrun condition occurs. Refer to “Section 30.4.5.1, Virtual FIFO Overun Smoother” on page 11for details.	CPU
0X45	0–7	OV_SkipFrame_Cnt	Internal variable. Set to zero	CPU
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 30.5.2.2, “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 located in internal Multiuser RAM. See Section 30.5.3.10, “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

Table 30-36. RX Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
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
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 and the Last PCD is reached. 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	Res	Initialize to zero.	CPU
0x58	0–31	L2QT	VLAN priority mapping table. See Section 30.5.3.11.1, “Rx Layer 2 QoS Table - L2QT.”	CPU
0x5C	32 Bytes	L3QT	IP priority mapping table. See Section 30.5.3.11.2, “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 30.5.3.8, “Address Filtering (AF) Field Description” for more details.	CPU

Table 30-36. RX Global Parameter RAM (continued)

Offset	Bits	Name	Description	Initialized by
0xC0		EXPGlobalParam	Base Address for Extended Parsing Global Parameters. See Section 30.6.1, "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 located in internal Multiuser RAM. See Section 30.4.8.1.1, "Loss Less Flow Control." This base address must be four bytes aligned. This base address must be programmed if Lossless Flow Control is enabled in REMODER[LossLessFCEn] OR if Advanced Queue Management is enabled by programming TAD[IWCT Index]>0. If Lossles Flow Control feature is enabled, the user must allocate 8..64 bytes for this data structure depending on the number of queues.	
0xF8-0xFF		Zero	Initialize to zero.	QE

30.5.3.6 Rx Thread Parameter RAM

The RX thread parameter RAM is summarized by the following table:

Table 30-37. 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

30.5.3.7 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	Res		Res	Res		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	Res	Loss Less FCEn	Res	RFSE	EXP	NumOfQueues			Reserved		VFSEn	Res	DXE	DNE	IPCHK	IPAE
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 30-37. UCC Ethernet Mode Register (REMODER)

Table 30-38. 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 30.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 30.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers. EXF=1 - Five individual addresses as programmed in Section 30.5.1.15, “Station Address Part 1 Register (MACSTNADDR1)” and Section 30.5.1.16, “Station Address Part 2 Register (MACSTNADDR2)” registers, and in Global Rx Parameter RAM AF entry as described in Section 30.5.3.8, “Address Filtering (AF) Field Description.” Refer to Figure 30-6 and Figure 30-7 for relationship between EXP and EXF fields.
1-3	Reserved	Set to zero.
4-5	Reserved	Set to zero

Table 30-38. REMODER Field Descriptions (continued)

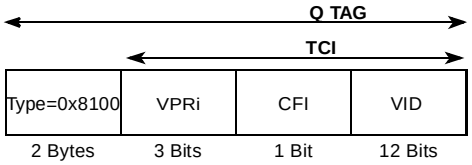
Bits	Name	Description
6-9	VTagOP	VLAN Operation for tagged income frames 0000 - No VLAN TAG operation is executed (NOP). 0001 - Replace VID portion of Q Tag. Other fields in TCI are left unchanged. 0010 - If VID=0 replace VID with default value. Other fields in TCI are left unchanged. 0011 - Extract 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 DA and the income 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
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 30.5.3.11.1, “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 30.4.12, “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	Advanced Queue Management : LossLess Flow Control Enable 0 - LossLess Flow Control feature is disabled 1 - LossLess Flow Control feature is enabled. See description of LossLessFCPtr in UEC Global Parameter RAM for more details.
18	Reserved	Set to zero
19	RFSE	Receive Firmware Statistics Counters enable (see Section 30.7.4, “Firmware Counters”.) 0 - Do not perform Firmware statistics on received frames. 1 - Perform Firmware statistics on received frames

Table 30-38. REMODER Field Descriptions (continued)

Bits	Name	Description
20	EXP	EXtended Address Parsing Mode 0 Extended Parsing mode disabled (MPC82xx-like filtering mode). Refer to Figure 30-6 for relationship between EXP and EXF fields. 1 Extended Parsing mode enabled. Refer to Figure 30-8 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
24-25	Reserved	Set to zero
26	VFSEn	Virtual FIFO Smoother enable 0 - Do not enable Virtual FIFO Smoother 1 - Enable Virtual FIFO Smoother See section “Section 30.4.5.1, Virtual FIFO Overun Smoother” on page 11 for more details.
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 30.7.2, “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 30.7.2, “Dynamic Minimum and Maximum Frame Length” for more details.
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.

30.5.3.8 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 30-38. Address Filtering (AF) Entry Definition

Table 30-39. 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 20.3.1, "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 QE 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.
0x3C-0x3F		Reserved	Set to zero.

30.5.3.9 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 30-39. Static Parameter (SP) Field Entry

Table 30-40. 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		

30.5.3.10 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 30-41. RxBD Parameter Table Description

Offset	Bits	Name	Description	Initialized by
Offset + 0	0:31	**PR0_BDBPTR**	BD ring0 Base pointer. The BD ring is located in external memory. RxBD ring0 base pointer. The internal BD ring includes 4 prefetched RxBDs.	QE
Offset + 4	0:31	**PR0_BDPTR**	RxBD ring0 pointer. RxBD ring0 pointer. Points to the next internal BD that the receiver transfers data. User writes: PR0_BDPTR	QE
Offset + 8	0:31	**PR0_EBDBPTR**	External BD ring0 Base pointer. The BD ring is located in external memory. External RxBD ring0 base pointer.	CPU
Offset + C	0:31	**PR0_EBDPTR**	External BD ring0 pointer. External RxBD ring0 pointer. Points to the next BD that the receiver transfers data. User writes:PR0_EBDPTR	QE
.....			Rest of BD Rings	CPU/QE
Offset + 70	0:31	**PR6_BDBPTR**	BD ring7 Base pointer. The BD ring is located in external memory. RxBD ring7 base pointer. The internal BD ring includes 4 perfected RxBDs.	QE
Offset + 74	0:31	**PR6_BDPTR**	RxBD ring7 pointer. RxBD ring7 pointer. Points to the next internal BD that the receiver transfers data. User writes: PR7_BDPTR	QE
Offset + 78	0:31	**PR7_EBDBPTR**	External BD ring7 Base pointer. The BD ring is located in external memory. External RxBD ring7 base pointer.	CPU
Offset + 7C	0:31	**PR7_EBDPTR**	External BD ring7 pointer. External RxBD ring7 pointer. Points to the next BD that the receiver transfers data. User writes:PR7_EBDPTR	QE

30.5.3.11 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 30-40), IP priority (See L3QT, Figure 30-41) or default priority. See QoS field in Section 30.5.3.7, "Rx Ethernet Mode Register (REMODER)."

30.5.3.11.1 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 30-40. Layer 2 QoS Table - L2QT

1 Offset from L2QT

Table 30-42. 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

30.5.3.11.2 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 30-41. Layer 3 QoS Table - L3QT

1 Offset from L3QT

Table 30-43. L3QT Description

Offset	Bits	Name	Description
Offset + 0	0:3	PR_0x0	If the original IP priority is equal to 0x0, it mapped to this field value.
	4:7	PR_0x1	If the original IP priority is equal to 0x1, it mapped to this field value.
. . .	. . .	. . .	. . .
Offset + F	8:11	PR_0x3E	If the original IP priority is equal to 0x62, it mapped to this field value.
	12:15	PR_0x3F	If the original IP priority is equal to 0x63, it mapped to this field value.

30.5.3.12 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 30-42. Rx Interrupt Coalescing Table

[1] Offset from IntCoalescingPTR

Table 30-44. 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 30-44. 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

30.5.4 Bus Mode Register (BMR*x*)

This register was named ‘FCR’ in MPC82xx.

Figure 30-43 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 30-43. Bus Mode Register (BMR*x*)

BMR*x* fields are described in Figure 30-45.

30.5.5 LossLess Flow Control

Table 30-45. 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	QE 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 QE, for debug purposes. See Figure 19-1and Section 5.4.3.7, “Debug Configuration.”
6	DTB	Indicates on what bus the data is located. 0 On the coherent system bus (CSB) 1 On the QE secondary bus. The programming of this bit must be consistent with the System Configuration.See Figure 19-1.
7	BDB	Indicates on what bus the BDs or are located. In Extended Parsing mode see also LookupBMR field in Section 30.6.2.6, “Hash Table Lookup PCDs” and Section 30.8.2, “Add/Remove Entry in Hash Lookup Table.” 0 On the coherent system bus (CSB). 1 On the QE secondary bus. The programming of this bit must be consistent with the System Configuration. See Figure 19-1.

See section Section 30.4.8.1.1, “Loss Less Flow Control,” for description of this feature.

Figure 30-44. 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															

Table 30-46. 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.

30.6 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 AF field is programmed to point to the first Parse Command Descriptor (PCD). Additional PCDs follow the first PCD in subsequent addresses.

30.6.1 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 30.6.2, “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 30-45. Extended Parsing Mode Global Parameters Definition

Table 30-47. 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

30.6.2 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 30-46. PC Opcodes

The following table specifies the PCD Opcodes:

Table 30-48. PCD Opcodes

Opcode Name	PCD Opcode	Comments
GenerateLookupKey_ EthFast	0x0	This PCD is used to generate a LookupKey from the L2 frame header fields. Since 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.

30.6.2.1 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							

Figure 30-47. Last PCD Field Descriptions

Offset + 4	Reserved
Offset + 6	

Figure 30-47. Last PCD Field Descriptions

The PCD fields are described in Table 30-50.

Table 30-49. Last PCD Field Descriptions

Offset	Bits	Name	Description
PCD_BASE+0	0-7	Opcode	0x3F
	8-15		Reserved bits. Must be cleared to zero.
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.

30.6.2.2 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	PCD ID
PCD_BASE+ 4	Reserved															
PCD_BASE + 6	Reserved															

Figure 30-48. Generate L2 LookupKey PCD

The PCD fields are described in Table 30-50.

Table 30-50. 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.
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 30.8.2.1, “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.

30.6.2.3 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 30-49. Change Mask PCD

The PCD fields are described in Table 30-50.

Table 30-51. 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.

30.6.2.4 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 30-50. Store LookupKey PCD

The PCD fields are described in Table 30-50.

Table 30-52. 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		

30.6.2.5 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 30-51. Restore LookupKey PCD

The PCD fields are described in Table 30-50.

Table 30-53. 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		

30.6.2.6 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).

30.6.2.6.1 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.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCD_BASE + 0	PCD OPCODE=0x20								LookupBMR							
PCD_BASE + 2	LookupKeySize								Reserved			EXT	HaskKeySize			
PCD_BASE + 4	LookupTableBase															
PCD_BASE + 6																

Figure 30-52. Four Way Hash Lookup PCD

The PCD fields are described in Table 30-50.

Table 30-54. 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 30.5.4, "Bus Mode Register (BMRx)" for details. Note that BDB bit detemines 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 (i.e. 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 (i.e. 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 30.6.2.6.3, "Four Way and Eight Way Hash Mode Lookup Table Entry (HLUT)" for details on this lookup algorithm.

30.6.2.6.2 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 30-53. Eight Way Hash Lookup PCD

The PCD fields are described in Table 30-50:

Table 30-55. 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 30.5.4, "Bus Mode Register (BMRx)" for details. Note that BDB bit detemines 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 (i.e. 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 (i.e. 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 30.6.2.6.3, "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.

30.6.2.6.3 Four Way and Eight Way Hash Mode Lookup Table Entry (HLUT)

The hash mode lookup table is split into two parts, termination action descriptors (TADs) and exact match tags. The table is organized into sets, which are lookup table entries indexed by the HashKey. Each set has four ways. Each way contains an 8- or 16-byte exact match tag that contains the LookupKey and an 8-byte TAD field. 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 comparison of all four tags yields 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 located in the PCD.

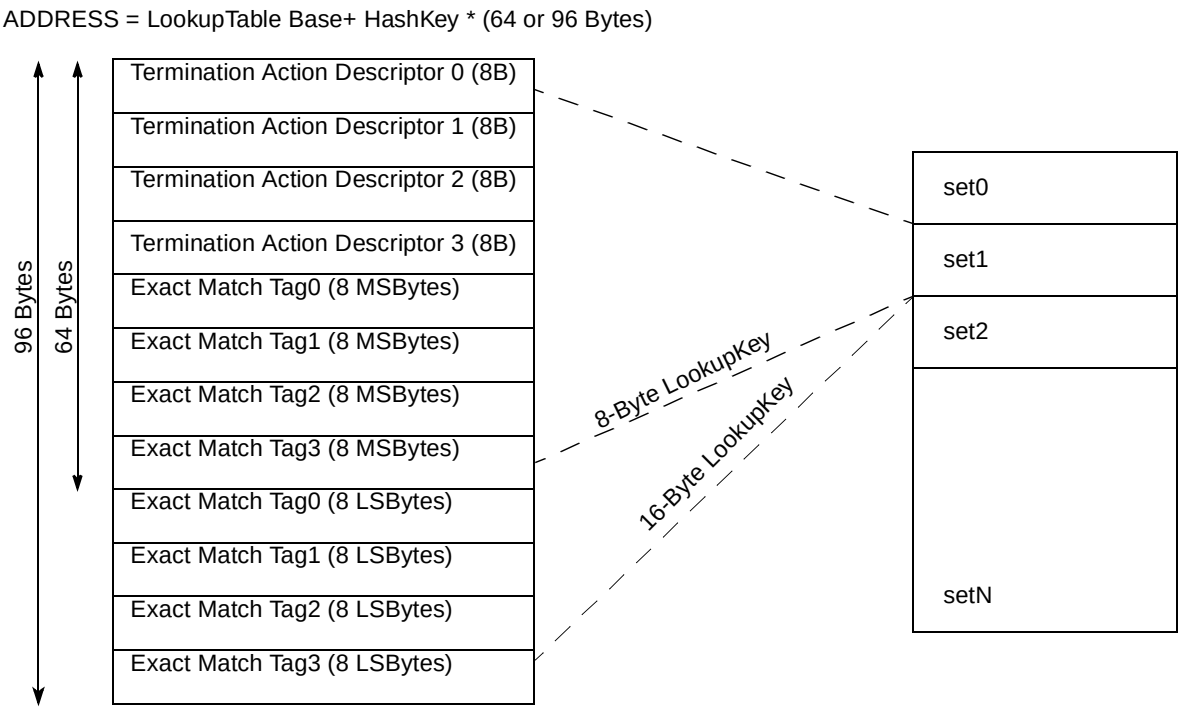


Figure 30-54. 4-Way Hash Mode Lookup Table Format

30.6.2.6.4 Exact Match Tags

The exact Match Tag is 8 or 16 bytes long, according to the LookupKeySize. See Section 30.14, " Exact Match Tags Memory Organization" for details.

30.6.2.6.5 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.

	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																

Figure 30-55. Termination Action Descriptor (TAD)

Table 30-56. TAD Field Descriptions

Offset	Bits	Name	Description
	0	EXF	EXtended Feature Mode (for termination). 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. See description in REMODER register. Section 30.5.3.7, “Rx Ethernet Mode Register (REMODER)
	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 1 - Reject Frame For proper operation, if TAD[Rej]=1 the user must program bits TAD[VPriority,CFI,VID]=0
	3	—	Reserved. Clear to zero.
0x0	6-15		See description in REMODER register. Section 30.5.3.7, “Rx Ethernet Mode Register (REMODER)
0x2	0-15	QTag	QTag (VPriority,CFI,VID) to be inserted.
0x4		Reserved	Set to zero

30.7 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.

30.7.1 Features

- Supports IEEE Std. 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 Std. 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

30.7.2 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 Std. 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” (that is, 64). In figure 3-3 there “Tagged Frame Format” IEEE shows MAC client data excluding Q-Tag (i.e. 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.

30.7.3 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

30.7.4 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. Since the atomicity of this operation is not guaranteed, the user must read the counter and verify that its value is zero or slightly larger.

30.7.4.1 TX Firmware Counters

Table 30-57. 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. (30.3.1.1.3)
TESBASE+4	MulColTx	32	M: Multiple Collision. Number of Frames that where involved in more than one collision and than transmitted succesfully (30.3.1.1.4)
TESBASE+8	LateColTxFr	32	Late collision. Counts all frames that were involved in late collision event during frame transmission (30.3.1.1.10)
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 (30.3.1.1.11)
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 e.g. OV/UN (30.1.1.12)
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 (30.3.1.13)
TESBASE+18	FrTxOK	32	M: Number of frames transmitted OK (30.3.1.1.2)
TESBASE+1C	TxFrExcessiveDiffer	32	Conts frames that their differal time was greater than a specified threshold. May (3.1.1.20)
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.

30.7.4.2 RX Firmware Counters

Table 30-58. 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. (30.3.1.1.6)
RESBASE+4	FrAlignEr	32	M:Alignment Errorrs Counter. Number of frames received that had Alignment errors (30.3.1.1.7). 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 (30.3.1.1.23)
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 dont care value in the case of a Jumbo frame) (3.1.1.24)
RESBASE+10	FrTooLongRx	32	Counts all received frames with length greater than the programmable parameter MaxLength (see above) (3.1.1.25)
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 (30.4.3.1.13)
RESBASE+1C	SymbolErrorRx	32	Number of received frames in which a received error symbol is reported from the PHY layer device during frame reception (30.4.3.1.17)
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 transfered to upper layers

Table 30-58. RX Firmware Counters (continued)

Address EtherStatsBase= RESBASE	Name	Width (bit)	Description
REBASE+30	EtherStatsUnderPkts	32	Total number of frames received that were less than 64 octets long but are a good frames otherwise (i.e. except from the error of being too short, they are good frames)
REBASE+34	EtherStatsPkts256	32	Total number of frames (Including bad frames) received that were between 256 (Including FCS length==4) and 511 octets . If MACCFG2[CRC]=1 in and the additional 4 bytes for the CRC cross the boundary between 511 and 512 bytes, the RMON counter is incremented for 256 bytes instead of for 512 bytes.
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 occured during reception, that is not counted on any other error counter. Counts all frames that were lost due to any error (e.g. OV) (30.3.1.1.15)
REBASE+48	PausFrRx	32	Counts the total number of PAUSE frames received by thi sMAC (30.3.4.3)
REBASE+4C	Reserved		
REBASE+50	RxRVLANcnt	32	Counts the total number of fames that their VLAN tag was removed.
REBASE+54	RxRepVLANcnt	32	Counts the total number of fames that their VLAN tag was replaced.
REBASE+58	RxInVLANcnt	32	Counts the total number of fames that VLAN tag was inserted to their header.
REBASE+5C	RxIPChcksumErr	32	Counts the total number of fames that have an IPChecsum Error.

30.7.5 UCC Statistics (Hardware Counters)

This section defines the MIB items located at the UCC.

Table 30-59. UCC Statistics

Address	Name	Standard Name	Width (bit)	Description
UCC*x*_BASE+ 0x180	TX64	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
UCC*x*_BASE+ 0x184	TX127	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
UCC*x*_BASE+ 0x188	TX255	TxPkts128	32	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
UCC*x*_BASE+ 0x18C	RX64	EtherStatsframe64	32	Total number of frames received including bad frames that were exactly in the minimal length (64 bytes)
UCC*x*_BASE+ 0x190	RX127	EtherStatsPkts65	32	Total number of frames (Including bad frames) received that were between 65bytes (Including FCS length==4) and 127 octets
UCC*x*_BASE+ 0x194	RX255	EtherStatsPkts128	32	Total number of frames (Including bad frames) received that were between 128 (Including FCS length==4) and 255 octets
UCC*x*_BASE+ 0x198	TXOK	OcTxOK	32	Octet Transmitted OK. Total number of octet residing in frames that where involved succesful transmission (30.3.1.1.8)
UCC*x*_BASE+ 0x19C	TXCF	PausFrTx	32	Counts the total number of PAUSE control frame transmitted by this MAC (30.3.4.2)
UCC*x*_BASE+ 0x1A0	TMCA	MulCastFrTxOK	32	Multicast Frame transmitted OK. Counts all frames that were transmitted succesfully with the group address bit set that are not broadcast frames (30.1.1.18). Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.
UCC*x*_BASE+ 0x1A4	TBCA	BroadCastFrTxOK	32	Broadcast frames that transmitted OK. Counts all frames transmitted succesfully that had DA field equals to the broadcast address (30.3.1.1.19). Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.

Table 30-59. UCC Statistics (continued)

Address	Name	Standard Name	Width (bit)	Description
UCC*x*_BASE+ 0x1A8	RXFOK	FrRxOK	32	Number of frames received OK (30.3.1.1.5)
UCC*x*_BASE+ 0x1AC	RBYT	OCRxOK	32	Octets received OK. Counts the total number of octets received OK in this probe. (30.1.1.14)
UCC*x*_BASE+ 0x1B0	RXBOK	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
UCC*x*_BASE+ 0x1B4	RMCA	MulCastFrRxOK	32	Multicast Frame Received OK. Counts all frames that were received succesfully with the group address bit set that are not broadcast frames (3.1.1.21). Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.
UCC*x*_BASE+ 0x1B8	RBCA	BroadCastFrRxOK	32	Broadcast frames that received OK. Counts all frames received succesfully that had DA field equals to the broadcast address (3.1.1.22). Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.
UCC*x*_BASE+ 0x1BC-0x1C0	SCAR/SCAM	SCAR/SCAM/CMR	32	These registers holds the carry bits and mask of the HW counters. See Section 30.7.7.17, “Statistic Counters Carry Register (SCAR)” and Section 30.7.7.18, “ Statistic Counters Mask Register.”
UCC*x*_BASE+ 0x1C4	RxDiscOV	RxDiscOV	32	Frames discarded due to overrun condition in the RxFIFO.

30.7.6 SW Statistics

The following MIB items can be implemented in SW by the CPU. The QE terminates the control frames. PAUSE control frames are terminated by the MAC, and statistics is gather by the QE. Non-PAUSE control frames are sent to the CPU termination queues.

Table 30-60. 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)

Table 30-60. MIB (continued)

Address	Name	Width (bit)	Description
SW	UnSprtOpcdRx	32	Counts all received control frames (T field = CONTROL) with op-code field that un suported by this device, including non-valid op-code field, and that are trasfered 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 calcuilated by the IEEE item BroadCastFrRxOK
SW	EtherStatsMCSTPkts	32	Total number of frames received that were directed to a MCST address. Can be calcuilated 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 succesfully (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

30.7.7 Detailed Description of HW Statistics Counters

30.7.7.1 Total Transmit 64-byte Frame Counter

offset	0x180															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TX64[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	TX64[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-56. Total Transmit 64-byte Frame Counter(TX64)

Table 30-61. TX64 Field Descriptions

Bits	Name	Description
0-31	TX64	Total number of frames Transmitted including bad frames that were exactly 64-bytes

30.7.7.2 Total Transmit Frame 65 to 127 Byte Packet Counter

offset	0x184															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TX127[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	TX127[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-57. Total Transmit Frame 65 to 127 Byte Packet Counter(TX127)

Table 30-62. TX127 Field Descriptions

Bits	Name	Description
0-31	TX127	Total number of frames (Including bad frames) transmitted that were between 65 (Including FCS length==4) and 127 octets

30.7.7.3 Total Transmit Frame 128 to 255 Byte Packet Counter

offset	0x188															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved								TX255[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	TX255[8:23]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-58. Total Transmit Frame 128 to 255 Byte Packet Counter(TX255)

Table 30-63. TX255 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

30.7.7.4 Total Receive 64-byte Frame Counter

offset	0x18C															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RX64[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	RX64[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-59. Total Receive 64-byte Frame (RX64)

Table 30-64. RX64 Field Descriptions

Bits	Name	Description
0-31	RX64	Total number of frames received including bad frames that were exactly in the mninimal length (64 bytes)

30.7.7.5 Total Receive Frame 65 to 127 Byte Packet Counter

offset	0x190															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RX127[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	RX127[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-60. Total Receive Frame 65 to 127 Byte Packet Counter(RX127)

Table 30-65. RX127 Field Descriptions

Bits	Name	Description
0-31	RX127	Total number of frames (Including bad frames) received that were between 65 (Including FCS length==4) and 127 octets

30.7.7.6 Total Receive Frame 128 to 255 Byte Packet Counter

offset	0x194															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved								RX255[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	RX255[8:23]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-61. Total Receive Frame 128 to 255 Byte Packet Counter (RX255)

Table 30-66. RX255 Field Descriptions

Bits	Name	Description
0-7	-	Reserved
8-31	RX255	Total number of frames (Including bad frames) received that were between 128 (Including FCS length==4) and 255 octets

30.7.7.7 Transmit Octet OK Counter

offset	0x198															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TXOK[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	TXOK[15:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-62. Transmit Octet OK Counter (TXOK)

Table 30-67. TXOK Field Descriptions

Bits	Name	Description
0-31	TXOK	Octet Transmitted OK. Total number of octet residing in frames that where involved succesfull transmission

30.7.7.8 Transmit Pause Frame Counter

offset	0x19C															
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	TXCF[0:15]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-63. Transmit Pause Frame Counter (TXCF)

Table 30-68. TXCF Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16-31	TXCF	Counts the total number of PAUSE control frame transmitted by this MAC

30.7.7.9 Transmit Multicast Frame Counter

offset	0x1A0															
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 30-64. Transmit Multicast Frame Counter (TMCA)

Table 30-69. TMCA Field Descriptions

Bits	Name	Description
0-31	TMCA	Multicast Frame transmitted OK. Counts all frames that were transmitted succesfully with the group address bit set that are not broadcast frames

30.7.7.10 Transmit Broadcast Frame Counter

offset	0x1A4															
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 30-65. Transmit Broadcast Frame Counter (TBCA)

Table 30-70. TBCA Field Descriptions

Bits	Name	Description
0-31	TBCA	Broadcast frames that transmitted OK. Counts all frames transmitted succesfully that had DA field equals to the broadcast address

30.7.7.11 Frame Receive OK Counter

offset	0x1A8															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXFOK[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	RXFOK[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-66. Frame Receive OK Counter (RXFOK)

Table 30-71. RXFOK Field Descriptions

Bits	Name	Description
0-31	RXFOK	Number of frames erceived OK

30.7.7.12 Octet Receive OK Counter

offset	0x1B0															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RBYT[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	RBYT[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-67. Octet Receive OK Counter (RBYT)

Table 30-72. RBYT Field Descriptions

Bits	Name	Description
0-31	RBYT	Octets received OK. Counts the total number of octets received OK in this probe.

30.7.7.13 Receive Total Number Octets Counter

offset	0x1AC															
Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXBOK[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	RXBOK[16:31]															
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-68. Receive Total Number Octets Counter (RXBOK)

Table 30-73. RXBOK Field Descriptions

Bits	Name	Description
0-31	RXBOK	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 octetts only good frames. It has to be implemented in HW because it includes octets in frames that never even reach the UC

30.7.7.14 Receive Multicast Frame Counter

offset	0x1B4															
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_000															

Figure 30-69. Receive Multicast Frame Counter (RMCA)

Table 30-74. RMCA Field Descriptions

Bits	Name	Description
0-31	RMCA	Multicast Frame Received OK. Counts all frames that were received succesfully with the group address bit set that are not broadcast frames

30.7.7.15 Receive Broadcast Frame Counter

offset	0x1B8															
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 30-70. Receive Broadcast Frame Counter (RBCA)

Table 30-75. RBCA Field Descriptions

Bits	Name	Description
0-31	RBCA	Broadcast frames that received OK. Counts all frames received succesfully that had DA field equals to the broadcast address

30.7.7.16 Receive Discard Overrun Counter (RxDiscOV)

offset	0x1C4															
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	RxDiscOV[16:31]															
R/W	R/W															
Reset	0000_0000_0000_0000															

Figure 30-71. Receive Discard Overrun Counter (RxDiscOV)

Table 30-76. RxDiscOV Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RxDiscOV	Receive frames which are discarded due to overrun condition in Rx FIFO.

30.7.7.17 Statistic Counters Carry Register (SCAR)

offset	0x1BC															
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	CROV
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_000															

Figure 30-72. Statistic Counters Carry Register (SCAR)

Table 30-77. SCAR 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
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	CROV	Carry Register Counter RxDiscOV bit
16-31	—	Reserved

30.7.7.18 Statistic Counters Mask Register

offset	0x1C0															
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	MT64	MT65	MT128	MR64	MR65	MR128	MOTOK	MTPF	MTMC	MTBC	MRFOK	MROK	MRTO	MRMC	MRBC	MROV
R/W	R/W															
Reset	0000_0000_0000_000															

Figure 30-73. Statistic Counters Mask Register (SCAM)

Table 30-78. SCAM Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16	MT64	Mask Register Counter TTPML Carry bit Mask
17	MT65	Mask Register Counter TXP65 Carry bit Mask
18	MT128	Mask Register Counter TXP128 Carry bit Mask
19	MR64	Mask Register Counter TRPML Carry bit Mask
20	MR65	Mask RegisterCounter RXP65 Carry bit Mask
21	MR128	Mask Register Counter RXP128 Carry bit Mask
22	MOTOK	Mask Register Counter OcTxOK Carry bit Mask
23	MTPF	Mask Register Counter TXPF Carry bit Mask
24	MTMC	Mask Register Counter TMCA Carry bit Mask
25	MTBC	Mask RegisterCounter TBCA Carry bit Mask
26	MRFOK	Mask Register Counter FrRxOK Carry bit Mask
27	MROK	Mask Register Counter OCRxOK Carry bit Mask
28	MRTO	Mask Register Counter RxTOC Carry bit Mask
29	MRMC	Mask Register Counter RMCA Carry bit Mask
30	MRBC	Mask Register Counter RBCA Carry bit Mask
31	MROV	Mask Register Counter MROV Carry bit Mask

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]

30.8 Ethernet Command Set

The transmit and receive commands described in the following sections are issued to the CECR. Please refer to Section 20.3.1, “QUICC Engine Command Register (CECR)” for detailed description of the command format.

The commands are described in Table 30-79.

Table 30-79. 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. TheSQQD[BDRConsumerAddress] 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 SQQD[BDRConsumerAddress] points to the beginning of the collided frame with TxBD[R] still set (the frame looks as if it was never sent).
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 SQQD[BDRConsumerAddress] 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 20.3.1.1, “QUICC Engine Commands.”

Receive commands are described in Table 30-80.

Table 30-80. 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.

Table 30-80. Receive Commands (continued)

Command	Description
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.

30.8.1 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 Transmit 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 20.3.2, “QUICC Engine Command Data Register (CECDR)”section 4.4.2). The data structure is 56 bytes long, and is used by the QE 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 20.3.1, “QUICC Engine Command Register (CECR)” for details.

The SBC (Sub Block Code) used in the CECR is taken from Table 20-3, 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 30-81. 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 30.4.4, “Multithreading” for an explanation.

The InitEnet fields are described in Table 30-81.

Table 30-81. 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.
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

Table 30-81. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
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 30.8, "Ethernet Command Set" for more details. 0000 - Suggested value for maximum Rx nominal bit rate of 1000/100/10Mbps (4 threads) 0001 - Suggested value for maximum Rx nominal bit rate of 10/100bps (1 thread)0010 - 2 thread 0011 - 6 threads 0100 - 8 threads
	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 30.8, "Ethernet Command Set" for more details. 0000 - Suggested value for maximum Tx nominal bit rate of 1000/100/10Mbps (4 threads) 0001 - Suggested value for maximum Tx nominal bit rate of 10/100bps (1 thread) 0010 - 2 thread 0011 - 6 threads 0100 - 8 threads
	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 -RISC 2 0011 -RISC1 and RISC2. Suggested value.
CECDR+c	0-7	Rx_Global_SNUM	SNUM of a Global task chosen for UEC Receiver. Refer to Table 20-15. One of the SNUMs is allocated for this thread; each SNUM is used once in the system. This entry must always be initialized.
	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.

Table 30-81. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+10 This entry must be always initialized for at least one thread	0-7	Rx Th1_SNUM	SNUM of a thread chosen for UEC Receiver. 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. This entry must be always initialized.
	8-25	Rx 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 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	Optional thread. 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	Optional thread. 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	Optional thread. 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	Optional thread. Suggested value is 0x3. See description in entry for Rx Global Parameter RAM.
CECDR+20	0-7	Rx Th5_SNUM	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.

Table 30-81. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+24	0-7	Rx Th6_SNUM	Optional thread. 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	Optional thread. 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	Optional thread. 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.
CECDR+3c This entry must be always initialized for at least one thread	0-7	Tx Th1_SNUM	SNUM of Transmitter of the UCC running the UEC. Refer to Table 20-15. One of the threads is allocated; each thread is used once in the system. This entry must always bre initialized.
	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.

Table 30-81. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+40	0-7	Tx Th2_SNUM	Optional thread. Refer to description of Rx 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	Optional thread. Refer to description of Rx 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	Refer to description of Rx Th1_SNUM
	8-25	Thread4 Parameter RAM Page	Optional thread. 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	Refer to description of Rx Th1_SNUM
	8-25	Thread5 Parameter RAM Page	Optional thread. 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+50	0-7	Tx Th6_SNUM	Optional thread. Refer to description of Rx 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.

Table 30-81. InitEnet Command Parameter (continued)

Offset	Bits	Name	Description
CECDR+54	0-7	Tx Th7_SNUM	Optional thread. Refer to description of Rx 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	Refer to description of Rx Th1_SNUM
	8-31	Thread8 Parameter RAM Page	Optional thread. 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

30.8.1.1 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.

30.8.2 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 20.3.1, “QUICC Engine Command Register (CECR)” and Section 20.3.1.1, “QUICC Engine Commands”), and to CECDR registers (see Section 20.3.2, “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 20-3. The CECDR register contains the base address in multiuser RAM where the data structure in Figure 30-74 resides. See Section 30.6.2.6, “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 20.4.4, “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 30-74. Set entry in Hash Lookup Table Command Parameters

Table 30-82. 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 30.5.4, “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 30.6.2.6, “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 (i.e. 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 30-82. 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.

30.8.2.1 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 GenerateL2 LookupKey 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 the following table:

Table 30-83. Source Port Values (SNUMs)

Source Port	SNUM
UCC1	0x01
UCC2	0x11
UCC3	0x21
UCC4	0x31

Table 30-83. Source Port Values (SNUMs)

Source Port	SNUM
UCC5	0x81
UCC6	0x91
UCC7	0xa1
UCC8	0xb1

Note that these are the SNUM the values as in 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.

30.9 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.

30.10 External Signal Description

30.10.1 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. The second is referred to as the GMII option and is a superset of the MII signals. The GMII option supports the 1000 Mbps Ethernet rate and TBI interface which shares pins with the GMII interface signals. Other interface options are the RMII which is a reduced pin implementation of the MII interface, the RGMII which is a reduced pin implementation of the GMII interface, and the RTBI which is a reduced pin implementation of the TBI interface. Table 30-84 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.”See Chapter 3, “Signal Descriptions.”

Table 30-84. Signal Properties

Name	Function	I/O	Pull Up
RX_CLK	MII —receive clock GMII, TBI,RGMII,RTBI,RMII—not in use	Input	—
GRX_CLK	GMII, RGMII—receive clock TBI,RTBI—PMA receive clock 0 (125Mhz or 62.5Mhz) MII,RMII—not in use	Input	—
RX_CLK1	TBI—PMA receive clock 1(second receive TBI 62.5Mhz clock) GMII,MII,RGMII,RTBI,RMII—not in use	Input	—
COL	MII—collision GMII, RMII, RGMII, RTBI—not in use	Input	—
CRS	MII—carrier sense TBI—signal detect (SDET) GMII,RMII,RGMII,RTBI—not in use	Input	—
RX_DV	G/MII—RX_DV TBI—receive code group (RCG[8]) bit 8 RMII—carrier sense data valid (CRS_DV) RGMII—(RX_CLK rising edge) - receive data valid RGMII—(RX_CLK falling edge)- receive error RTBI—(RX_CLK rising edge) receive code group(RCG[4]) bit 4 RTBI—(RX_CLK falling edge) receive code group(RCG[9]) bit 9	Input	—
RXD[7:4]	GMII—receive data bits 7:4 (RXD[7:4]) TBI—receive code group (RCG[7:4]) bits 7:4 MII, RMII, RGMII, RTBI—not in use	Input	—
RXD[3:0]	GMII,MII—receive data bits 3:0, (RXD[3:0]) TBI—receive code group (RGC[3:0]) bits 3:0 RGMII—RX_CLK rising edge - RXD[3:0] RGMII—RX_CLK falling edge - RXD[7:4] RTBI—RX_CLK rising edge - RCG[3:0] RTBI—RX_CLK falling edge - RCG[8:5] RMII—receive data 1:0 (RXD [1:0])	Input	—
RX_ER	GMII,MII—receive error (RX_ER) RMII— receive error (RX_ER) TBI—receive code group (RCG[9]) RGMII, RTBI—not in use	Input	—
TX_CLK	MII—TX_CLK (transmit clock) RMII—REF_CLK (reference clock) This signal usually comes from an external oscillator or PHY GMII,RGMII,TBI,RTBI-oscillator source for transmit clock	Input	Passive
GTX_CLK	GMII/TBI - continuos transmit clock feedback RGMII/RTBI-inverted transmit clock feedback	Output	Passive
TXD[4:7]	GMII - transmit data bits 7:4 (TXD[7:4]) TBI—transmit code group (TCG[7:4]) bits 7:4 RMII, RGMII, RTBI—not in use	Output	Passive

Table 30-84. Signal Properties (continued)

Name	Function	I/O	Pull Up
TXD[3:0]	GMII,MII - transmit data bits 3:0 TBI- transmit code group (TCG[3:0]) bits 3:0 RMII—transmit data bits 1:0 RGMII—(TX_CLK rising edge) (transmit data bits 3:0) RGMII—(TX_CLK falling edge) (transmit data bits 7:4) RTBI—(TX_CLK rising edge) transmit code group (TCG[3:0]) bits 3:0 RTBI—(TX_CLK falling edge) transmit code group (TCG[8:5]) bits 8:5	Output	Passive
TX_EN	GMII,MII—TX_EN (transmit data enable) RMII—TX_EN (transmit data enable) TBI—transmit code group (TCG[8]) bit 8 RGMII—(TX_CLK rising edge) -transmit enable RGMII—(TX_CLK falling edge) -transmit error RTBI—(TX_CLK rising edge) - transmit code group (TCG[4]) bit 4 RTBI—(TX_CLK falling edge) - transmit code group (TCG[9]) bit 9	Output	Passive
TX_ER	GMII,MII- transmit error TBI—transmit code group (TCG[9]) bit 9 RMII, RGMII, RTBI—not in use	Output	Passive
Management signals	Refer to MII management signals. Management signals are common to all PHY interface.	Input/Output	Passive

30.10.2 Detailed Signal Descriptions

Table 30-85 provides the UCC Ethernet Controller interface signal descriptions.

Table 30-85. 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.
RX_CLK1	I		TBI—PMA receive clock 1(second receive TBI 62.5Mhz clock) GMII,MII,RGMII,RTBI,RMII—not in use
		State Meaning	
		Timing	Provides a timing reference for RCG[9:0] in TBI mode
GRX_CLK	I	Receive Clock from PHY. GMII, RGMII—receive clock from PHY TBI mode—Depending on the PHY being used, this is the input for a 62.5 MHz PMA receive clock 0 split phase with PMA_RX_CLK1 and is supplied by SERDES, or it is a 125 MHz receive clock. RTBI- PMA receive clock 0 works in 125 MHz receive clock	
		State Meaning	
		Timing	Provides a timing reference for RX_DV, RXD, and RX_ER or RCG[9:0] in TBI mode
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 30-85. 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 GMII,MII mode—When this signal is asserted, PHY is indicating that valid data is present on the G/MII interface. RGMII mode—Rising edge (RX_DV), falling edge (RX_ER) TBI mode—Represents receive code group (RCG[8]). Together with RCG[9] and RCG[0:7], they represents the 10-bit encoded symbol of GMII receive signals. RTBI mode—Rising edge (RCG[4]), falling edge (RCG[9]) 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 GMII,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. RGMII mode—Rising edge (RXD[3:0]), falling edge (RXD[7:4]) RMII—RXD[1:0] represents a di-bit of data to be transferred from PHY to MAC when CRS_DV is asserted. TBI mode—RXD[3:0] represents RCG[3:0]. Together with RCG[9:4], they represent the 10-bit encoded symbol of G/MII receive signals. RTBI mode—Rising edge (RCG[3:0]), falling edge (RCG[8:5])	
		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, GRX_CLK for GMII/RGMII/TBI/RTBI and TX_CLK in RMII
RXD[7:4]	I	Receive Data GMII—RXD[7:4] 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 TBI mode—RXD[7:4] represents RCG[7:4]	
		State Meaning	RXD is a bundle of data signals (RXD<7:4>). RXD<7:4> are driven by PHY.
		Timing	That transition synchronously with respect to GRX_CLK.

Table 30-85. Signal Descriptions (continued)

Signal	I/O	Description	
RX_ER	I	Receive Error GMII/MII and RMII mode—When RX_ER and RX_DV are asserted, PHY has detected an error in the current frame. TBI mode—Represents RCG[9]. Together with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.	
		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 and GRX_CLK for GMII/RGMII/TBI/RTBI
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 GMII mode- oscillator source for transmit clock RMII mode— reference clock to TX_EN, TXD, TX_ER, RXD, RX_ER and RX_DV Note that in GMII/RMII/RGMII/TBI/RTBI is not the reference clock to TX_EN,TXD and TX_ER	
		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. GMII/RGMII/TBI/RTBI - oscillator source for transmit clock
		Timing	—
TXD[7:0]	O	Transmit Data GMII mode—TXD[7:0] represent one complete octet of data to be sent from MAC to PHY when TX_EN is asserted and has no meaning when TX_EN is de-asserted. TBI mode—TXD[7:0] represents transmit code group (TCG) bits 7:0. Together with TCG[9:8], they represent the 10-bit encoded symbol or concatenation of G/MII transmit signals. 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. RGMII mode—Rising edge (transmit data bit 3:0), falling edge (transmit data bit 7:4) RTBI mode—Rising edge (TCG bit 3:0), falling edge (TCG bit 8:5)	
		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 and GTX_CLK in GMII/RGMII/TBI/RTBI modes

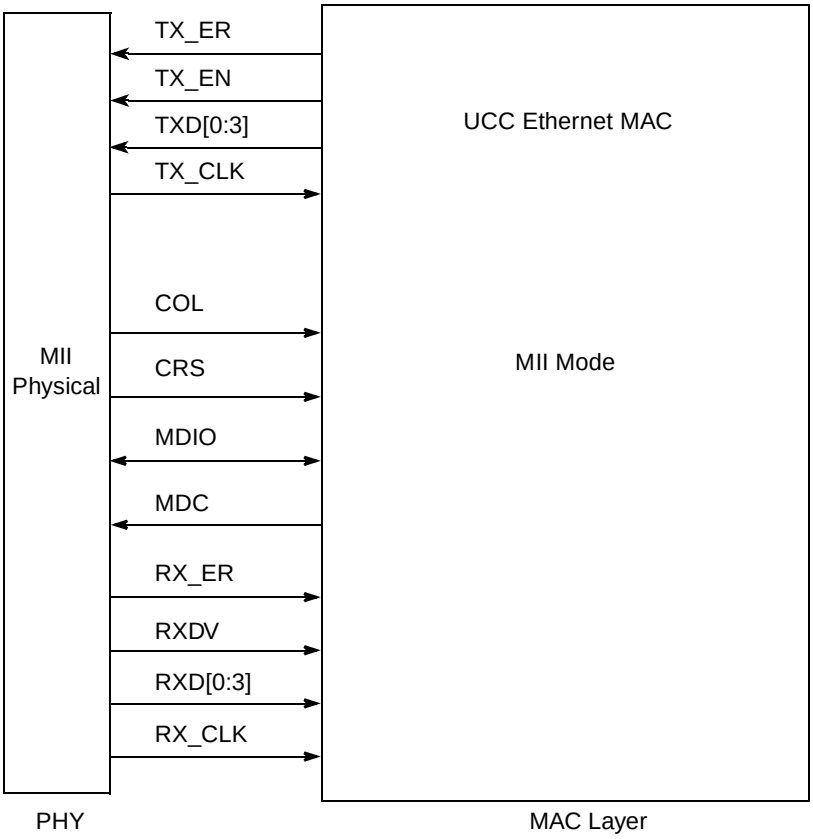
Table 30-85. Signal Descriptions (continued)

Signal	I/O	Description	
TX_EN	O	Transmit Enable GMII/MII/RMII mode—When this signal is asserted, MAC is indicating that valid data is present on the physical line RGMII - represent TX_EN on GTX_CLK rising edge and TX_ER on GTX_CLK falling edge RTBI - represent TCG[4] on GTX_CLK rising edge and TCG[9] on GTX_CLK falling edge TBI mode—represents TCG[8]. Together with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol or concatenation of GMII transmit signals.	
		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 GMII/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. TBI - represent transmit code group (TCG[9]) bit 9	
		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/GTX_CLK periods while TX_EN is also asserted,
MDC	O	Management data clock In GMII/MII/RGMII/RMII/TBI or RTBI 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 three-state circuits that enable either STA or PHY to drive the signal.

30.11 UCC Ethernet Controller Connections

30.11.1 Media Independent Interface (MII)

The UCC Ethernet Controller implements the IEE 802.3 standard MII interface for Fast Ethernet. Figure 30-75 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 the IEEE 802.3 standard (total:16 ports).

Figure 30-75. MII MAC-PHY Interface

Table 30-86 provides a list of the MII signals.

Table 30-86. MII Signal Descriptions

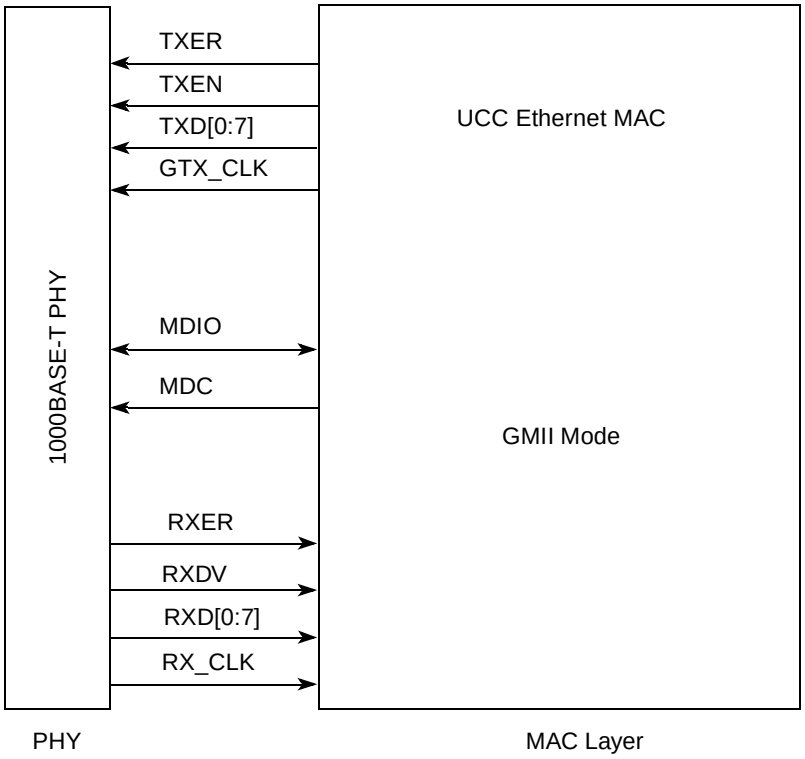
IEEE Name	I/O	Size	Function	Reference Clock
TX_ER	O	1	Transmit error, asserted by the MAC layer to generate a coding error on the byte currently being transferred over TXD[0:3].	TX_CLK
TX_EN	O	1	Transmit enable, asserted by the MAC sublayer when the first transmit preamble byte is driven over the MII. 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

Table 30-86. MII Signal Descriptions (continued)

IEEE Name	I/O	Size	Function	Reference Clock
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)	—

30.11.2 Gigabit Media Independent Interface (GMII)

The UCC Ethernet Controller implements the IEE 802.3 standard GMII interface for Gigabit Ethernet. Figure 30-76 describes the GMII interface signals and shows the basic components of the GMII including the signals required to make a UEC module connection with a PHY.



Note: Signal names are according to the IEEE 802.3 standard (total: 22 ports for full-duplex operation).

Figure 30-76. GMII MAC-PHY Interface

Table 30-87 provides a list of the GMII signals.

Table 30-87. GMII Signals

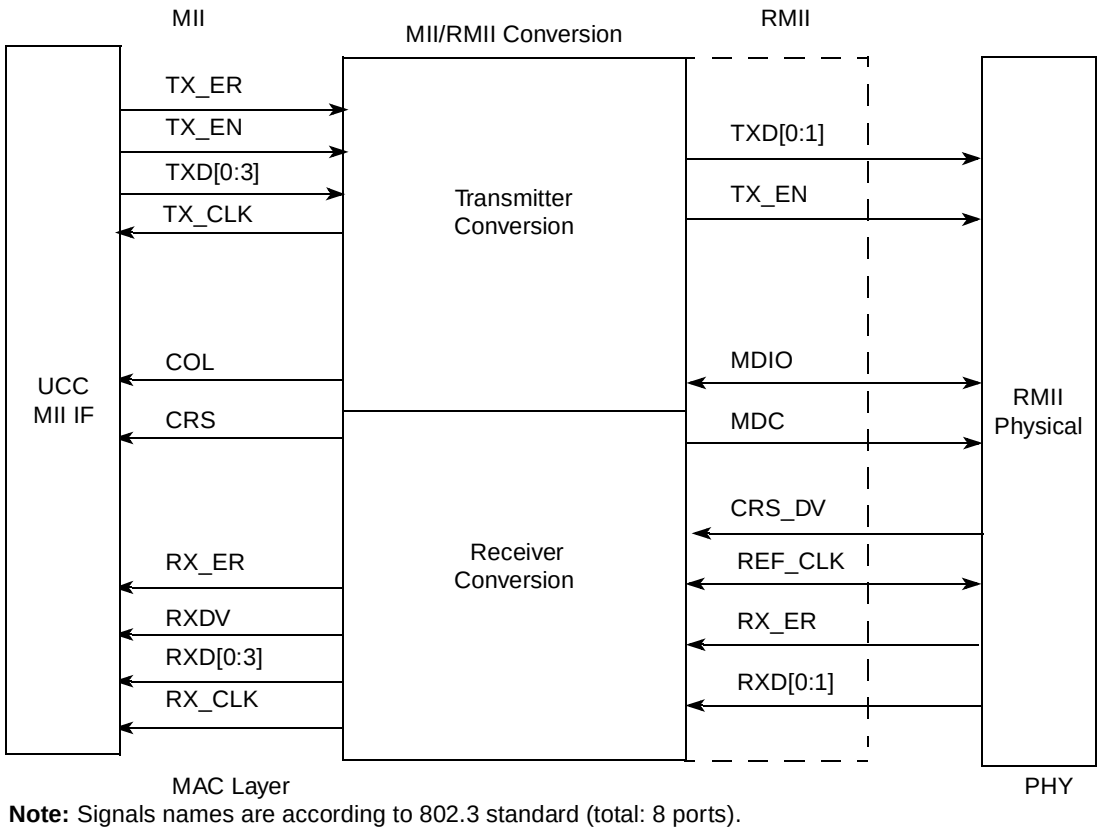
IEEE Name	I/O	Size	Function	Reference Clock	Timing
TX_ER	O	1	Transmit error, asserted by the MAC layer to generate a coding error on the byte currently being transferred over TXD[7:0].	GTX_CLK	max–2.5 n min–0.5 n
TX_EN	O	1	Transmit enable, asserted by the MAC sublayer when the first transmit preamble byte is driven over the GMII. It remains asserted for the remainder of the frame, up to the last CRC byte.	GTX_CLK	max–2.5 n min–0.5 n
TXD[0:7]	O	8	Transmit data	GTX_CLK	max–2.5 n min–0.5 n
GTX_CLK	O	1	125-MHz clock which provides the timing reference for the transfer of the TXEN, TXER, and TXD signals	—	—
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[7:0].	RX_CLK	max–2.5 n min–0.5 n

Table 30-87. GMII Signals (continued)

RX_DV	I	1	Receive data valid, asserted by the PHY layer when the first received preamble byte is driving over the GMII. It remains asserted for the remainder of the frame, up to the last CRC byte	RX_CLK	max–2.5 n min–0.5 n
RX_CLK	I	1	Receive clock, synchronized all receive signals (RX_DV, RXD, RX_ER).	—	—
RXD[0:7]	I	8	Receive data	RX_CLK	max–2.5 n min–0.5 n
MDIO	I/O	1	Management data input/output, used to transfer control signals between the PHY layer and the manager entity.	Management Clock	—
MDC	I	1	Management data clock, the MDIO signal clock reference (25 MHz clock)	—	—

30.11.3 Reduced Media Independent Interface (RMII)

The UCC Ethernet Controller MAC implements the MII/RMII Conversion to achieve pin-out reduce interface. Figure 30-77 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 30-77. RMII MAC-PHY Interface

Table 30-88 provides a list of the GMII signals.

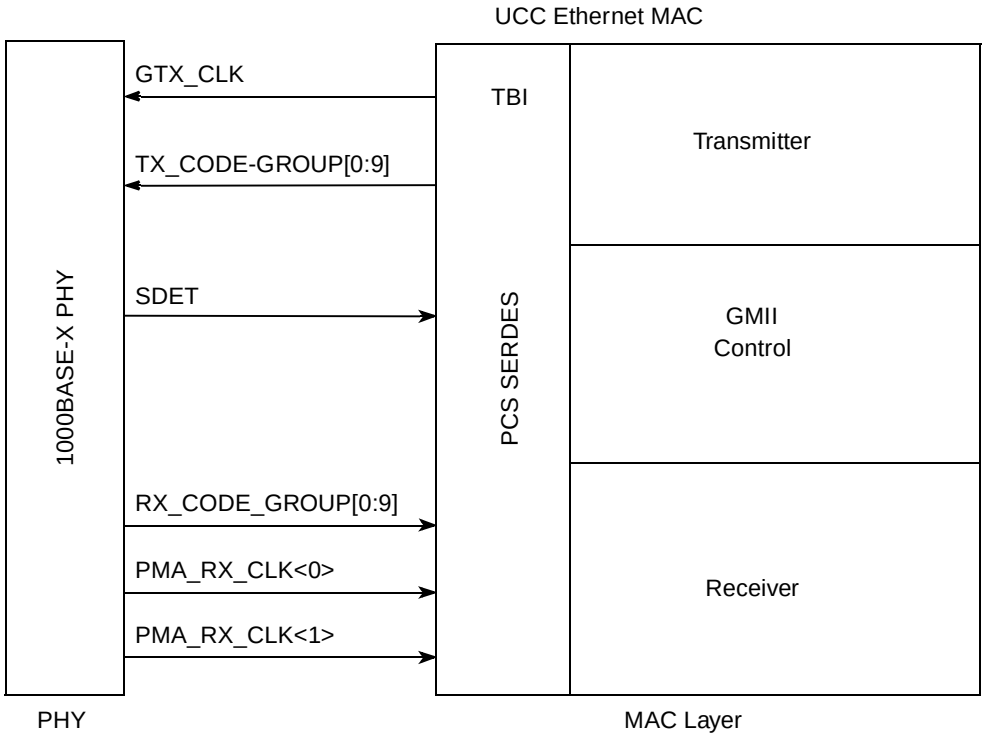
Table 30-88. RMII Signals

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
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)

30.11.4 TBI (Ten-Bit Interface)

Figure 30-78 shows the ten-bit interface (TBI) intended to be used between PHY and the UCC Ethernet Controller MAC to implement a standard SERDES interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 30-78 also shows the basic components of the TBI including the signals required to make a UEC module connection with a PHY. PMA_RX_CLK0 and PMA_RX_CLK1 are differential 62.5 MHz receive clocks.



Note: Signal names are according to 802.3 and ANSI standards (total: 24 ports).

Figure 30-78. TBI MAC-PHY Interface

Table 30-89 provides a list of the TBI signals.

Table 30-89. TBI Signals

Signal	I/O	Size	Function	Reference Clock	Timing
TCG[0:9]	O	10	Transmit data, driving symbol each clock	TX_CLK	max–2 n min–1 n
RCG[0:9]	I	10	Receive data, driving symbol each clock	PMA_RX_CLK	max–2 n min–1 n
TX_CLK	O	1	Transmit clock (125 MHz)	—	—
COM_DET	I	1	Comma detect	PMA_RX_CLK	max–2 n min–1 n
PMA_RX_CLK[0]	I	1	Receive clock 0—The 62.5-MHz recover receive byte clock. Used to latch odd numbered bytes of receive data.	—	—
PMA_RX_CLK[1]	I	1	62.5-MHz recover receive byte clock. This clock is 180° out-of-phase with PMA_RX_CLK<0>, and is used to latch even numbered bytes of receive data.	—	—

30.11.5 Reduced Gigabit Media Independent Interface (RGMII)

The RGMII is intended to be an alternative to the IEEE802.3u MII, the IEEE802.3z GMII, and the TBI. The principle objective is to reduce the number of pins required to interconnect the MAC and PHY from a maximum of 26 pins (GMII) to 13 pins (GTX_CLK125 included) in a cost effective and technology independent manner. In order to accomplish this objective, the data paths and all associated control signals will be reduced, control signals will be multiplexed together, and both edges of the clock will be used.

Figure 30-79 shows the basic components of the RGMII including the signals required to make UCC Ethernet Controller module connection with a PHY.

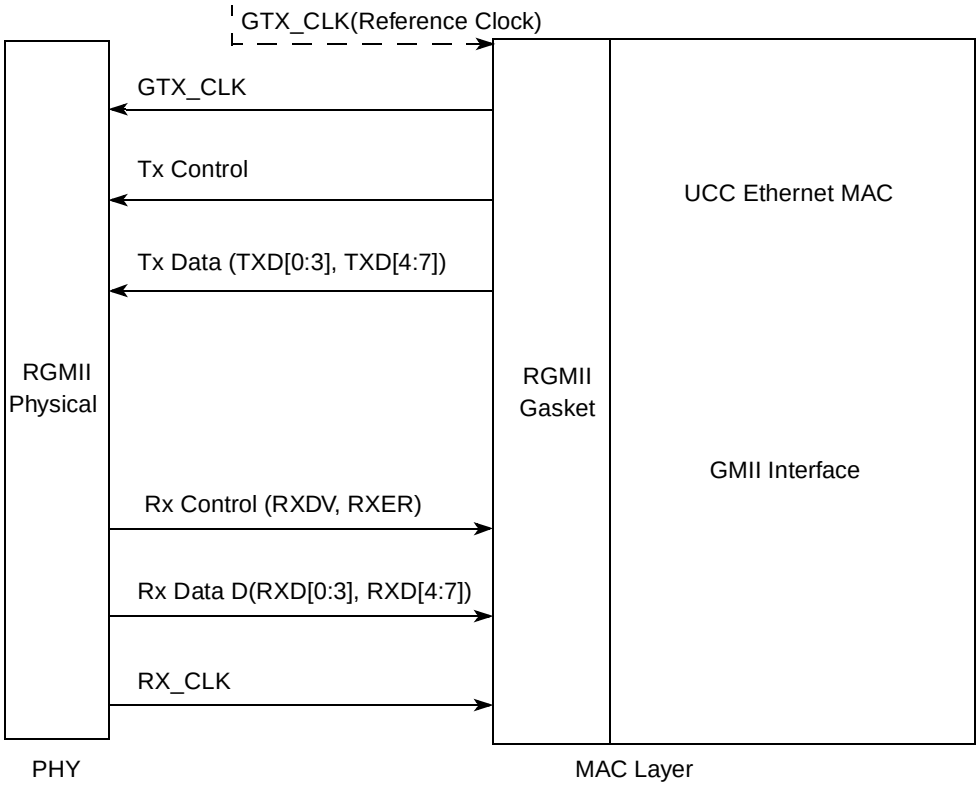


Figure 30-79. RGMII MAC-PHY Interface

Table 30-90 provides a list of the RGMII signals.

Table 30-90. RGMII Signals

IEEE Name	I/O	Size	Function	Reference Clock	Timing
TXC	I	1	Transmit reference clock will be 125 MHz	—	—
TD[0:3]	O	4	TXD[0:3] on clock posedge TXD[4:7] on clock negedge	TXC	max–2 n min–1 n
TX_CTL	O	1	TX_EN on clock posedge TX_ER on clock negedge	TXC	max–2 n min–1 n
RXC	I	1	Continuous receive reference clock will be 125 MHz	—	

Table 30-90. RGMII Signals (continued)

IEEE Name	I/O	Size	Function	Reference Clock	Timing
RD[0:3]	I	4	RXD[0:3] on clock posedge RXD[4:7] on clock negedge	RXC	max–2 n min–1 n
RX_CTL	O	1	RX_DV on clock posedge RX_ER on clock negedge	RXC	max–2 n min–1 n
MDIO	I/O	1	Management data input/output, used to transfer control signals between the PHY layer and the manager entity.	Management Clock	—
MDC	I	1	Management data clock, the MDIO signal clock reference (25 MHz clock).	—	—

30.11.6 Reduced Ten Bit Interface (RTBI)

An RTBI interface has 15 signals (GTX_CLK125 included), as defined by the RGMII specification Version 1.2a 9/22/00, and is intended to be an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI standard for connecting to an Ethernet PHY.

Figure 30-80 shows the basic components of the RGMII including the signals required to make UCC Ethernet Controller module connection with a PHY.

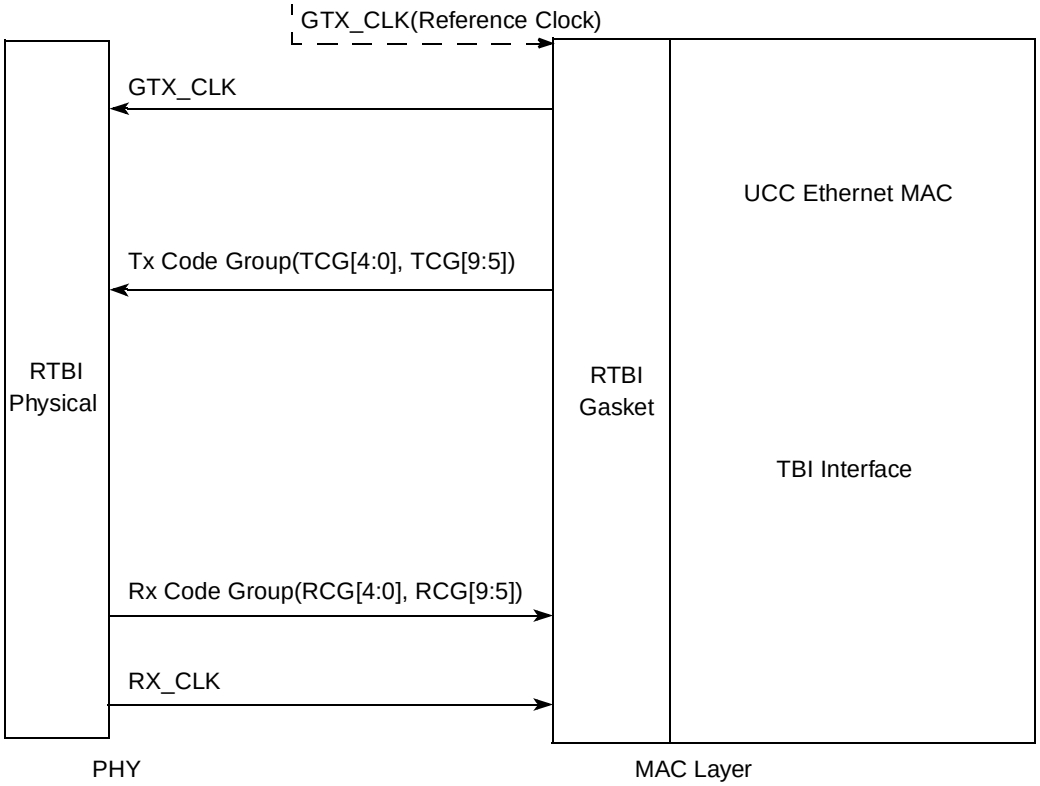


Figure 30-80. RTBI MAC-PHY Interface

Table 30-91 provides a list of the RGMII signals.

Table 30-91. RTBI Signals

IEEE Name	I/O	Size	Function	Reference Clock	Timing
TXC	I	1	Transmit reference clock will be 125 MHz	—	—
TCG[4:0]	O	4	TCG[4:0] on clock posedge TCG[9:5] on clock negedge	TXC	max–2 n min–1 n
RXC	I	1	Continuous receive reference clock will be 125 MHz	—	
RCG[4:0]	I	4	RXD[4:0] on clock posedge RXD[9:5] on clock negedge	RXC	max–2 n min–1 n

30.11.7 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers

30.11.7.1 TBI Transmit Process

The UEC's TBI Implements the Transmit portion of the Physical Coding Sublayer as found in Clause 36 of IEEE 802.3z. In SERDES mode, packets conveyed across the GMII are encapsulated and encoded into 10-bit symbols and output to the SERDES. In GMII mode, the GMII signals are passed through to the attached GMII PHY.

30.11.7.1.1 Packet Encapsulation

If TX_EN is de-asserted the UEC outputs an idle stream. If TX_EN asserts, a Start_of_Packet symbol is output. This symbol replaces the first byte of the preamble field. All other bytes of the packet pass through an 8B10B encoding module. After the last byte of the FCS field is signaled via the GMII, the MAC de-asserts TX_EN. The UEC then outputs an End_of_Packet symbol. Then, depending on the position of the End_of_Packet symbols (being in either an odd or even position) the UEC outputs one or two Carrier_Extend symbols. Following the last Carrier_Extend symbol, the UEC resumes sending idle codes. If, during a packet, the UEC wishes to mark a byte invalid, TX_ER is asserted. The UEC, upon detection of TX_ER, substitutes the data symbol for an Error_Propagation symbol.

30.11.7.1.2 8B10B Encoding

Every eight-bit data octet has two (not necessarily different) ten-bit symbols associated with it. Depending on the running disparity (the cumulative difference of ones and zeroes) the UEC module chooses the appropriate symbol.

Special encapsulation symbols are called ordered_sets. Ordered_sets are comprised of one to four ten-bit symbols. Ordered_sets can be found in Clause 36 of the IEEE 802.3z specification.

30.11.7.1.3 Preamble Shortening

Because the idle ordered_set comprises two symbols and begins on an even symbols boundary, packets can only begin on an even boundary. However, the GMII has no such restriction and may signal TX_EN

on an odd boundary. If this happens, the UEC delays the Start_of_Packet symbol, effectively ignoring the first byte of preamble; thus, a seven octet preamble becomes six octets on the Ten-Bit Interface.

30.11.7.2 TBI Receive Process

The UEC's TBI Implements the Receive portion of the Physical Coding Sublayer as found in Clause 36 of IEEE 802.3z. The Receive portion includes the Synchronization state machine. In SERDES mode, the UEC first attempts to acquire synchronization on the link by examining received symbols. Once synchronization is acquired, received packets are decoded and sent across the Receive GMII interface. In GMII mode, the GMII signals are passed through to the MAC.

30.11.7.2.1 Synchronization

The UEC examines received symbols looking for the seven bit 'comma' string embedded in some special symbols. Both the idle ordered_set and the Configuration ordered_set contain a symbol which has the comma. Once a certain number of codes with comma are detected, the TSEEC is considered to have acquired synchronization.

30.11.7.2.2 Auto-Negotiation for 1000BASE-X

Once synchronization is acquired, ordered_sets are decoded. If Configuration ordered_sets are received, the UEC decodes the two octet data field and the sixteen-bit Configuration data is stored and used to Auto-Negotiate with the link partner. in the Receive Configuration Register (RXCR[15:0]) an internal register used to receive all the link partners informations and used to compare to local ability during negotiation. Not visible to user.If, during Auto-Negotiation an invalid symbol is detected, Auto-Negotiation re-starts. After Auto-Negotiation is completed the TBI MII Status Register SR[AN done] in set. In this mode, packets may be received from the link partner.

30.11.7.3 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All of the TBI registers are 16 bits wide. The TBI registers are accessed at the offset of the TBI physical address. The UEC's TBI physical address is stored in the UTBIPA register. Writing to the TBI registers is performed in a way similar to writing to an external PHY; by using the MII management interface. By using UTBIPA in place of the PHY address, in the MIIMADD[PHY Address] field, and setting the MIIMADD[Register Address] to the appropriate address offset that corresponds to the register that one wants to read or write (see Table), the user can read (set MIIMCOM[read cycle]) or write (writing to MIIMCON[PHY control]) to the TBI block.

Table 30-92. TBI MII Register Set

Offset Address	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				
0x00	Control (CR)	R/W[1]	16 bits	30.11.7.4/30-142
0x01	Status (SR)	R, LH, LL	16 bits	30.11.7.5/30-143
0x02–0x03	Reserved	R	2 bytes	—

Table 30-92. TBI MII Register Set (continued)

Offset Address	Name	Access	Size	Section/Page
0x04	AN advertisement (ANA)	RW, R	16 bits	30.11.7.5/30-143
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	30.11.7.7/30-146
0x06	AN expansion (ANEX)	R, LH	16 bits	30.11.7.8/30-148
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	30.11.7.9/30-148
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	30.11.7.10/30-149
0x0F	Extended status (EXST)	R	16 bits	30.11.7.11/30-150
0x10	Jitter diagnostics (JD)	R/W	16 bits	30.11.7.12/30-151
0x11	TBI control (TBICON)	R/W	16 bits	30.11.7.13/30-152

[1] R = means Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

30.11.7.4 Control Register (CR)

Figure 30-81 describes the definition for the CR register.

	0	1	2	3	4	5	6	7	8	9	10					15
R	PHY Reset	0	Speed [0]	AN Enable	0	0	Reset AN	Full Duplex	0	Speed [1]	0	0	0	0	0	0
W																
Reset	0001_0001_0100_0000															
Offset	0x00															

Figure 30-81. Control Register Definition

Table 30-93 describes the fields of the CR register.

Table 30-93. CR Field Descriptions

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The control and status registers are returned to their default value. This in turn may change the internal state of the TBI and its link partner.
1	—	Reserved
2	Speed[0]	Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed. Setting this field controls the speed at which the TBI operates. The following table provides the appropriate encoding. Its default is bit[2]='0';bit[9]='1'
3	AN Enable	Auto-negotiation enable. This bit is set by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled.
4–5	—	Reserved

Table 30-93. CR Field Descriptions (continued)

Bits	Name	Description
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation. 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.
7	Full Duplex	Duplex mode. This bit is set by default. 0 Half-duplex operation. 1 Full-duplex operation.
8	—	Reserved, should be cleared.
9	Speed[1]	Speed selection. This bit defaults to a set state and should always be set, which corresponds to 1000 Mbps speed. **Maximum Operating Speed / Bit 2 / Bit 9** Reserved / 0 / 0 Reserved / 1 / 0 1000 Mbps / 0 / 1 Reserved / 1 / 1
10–15	—	Reserved

30.11.7.5 Status Register (SR)

Figure 30-82 describes the definition for the SR register.

	0						6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	Extend Status	0	No Pre	AN Done	Remote Fault	AN Ability	Link Status	0	Extend Ability
W																
Reset	0000_0001_0100_1001															
Offset	0x01															

Figure 30-82. Status Register Definition

Table 30-94 describes the fields of the SR register.

Table 30-94. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	Extend Status	This bit indicates that PHY status information is also contained in the Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, should be cleared.

Table 30-94. SR Descriptions (continued)

Bits	Name	Description
9	No Pre	MF preamble suppression enable. This bit indicates whether or not the PHY is capable of handling MII management frames without the 32-bit preamble field. Returns 1, indicating support for suppressed preamble MII management frames. This bit is read-only.
10	AN Done	Auto-negotiation complete. This bit is read-only and is cleared by default. 0 Either the auto-negotiation process is underway or the auto-negotiation function is disabled. 1 The auto-negotiation process has completed.
11	Remote Fault	Remote fault. This bit is read-only and is cleared by default. Each read of the status register clears this bit. 0 Normal operation. 1 A remote fault condition was detected. This bit latches high in order for software to detect the condition.
12	AN Ability	Auto-negotiation ability. While read as set, this bit indicates that the PHY has the ability to perform auto-negotiation. While read as cleared, this bit indicates the PHY lacks the ability to perform auto-negotiation. Returns 1 on read. This bit is read-only.
13	Link Status	Link status. This bit is read-only and is cleared by default. 0 A valid link is not established. This bit latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, should be cleared.
15	Extend Ability	Extended capability. This bit indicates that the PHY contains the extended set of registers (those beyond control and status). Returns 1 on read. This bit is read-only.

30.11.7.6 AN Advertisement Register (ANA)

Figure 30-83 describes the definition for the ANA register.

	0	1	2 3	4		6	7 8	9	10	11				15
R	Next Page	Ack	Remote Fault	0	0	0	Pause	Half Duplex	Full Duplex	0	0	0	0	0
W														
Reset	0000_0000_0000_0000													
Offset	0x04													

Figure 30-83. AN Advertisement Register Definition

Table 30-95 describes the fields of the ANA register.

Table 30-95. ANA Field Descriptions

Bits	Name	Description
0	Next Page	Next page configuration. The local device sets this bit to either request next page transmission or advertise next page exchange capability. 0 The local device wishes not to engage in next page exchange. 1 The local device has no next pages but wishes to allow reception of next pages. If the local device has no next pages and the link partner wishes to send next pages, the local device shall send null message codes and have the message page set to 0b000_0000_0001, as defined in annex 28C.

Table 30-95. ANA Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>1</td><td>Ack</td><td>Acknowledge. Write 0, ignore on read.This bit is read-only. (Reserved)</td></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The local device's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>The local device's PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult Table 30-96.
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability.
0 Designates local device as not capable of half-duplex operation.
1 Designates local device as capable of half-duplex operation.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability.
0 Designates the local device as not capable of full-duplex operation.
1 Designates the local device as capable of full-duplex operation.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

Table 30-96 describes the resolution of pause priority.

Table 30-96. PAUSE Priority Resolution

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	x	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	1	Enable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Enable PAUSE receive
1	0	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	0	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive
1	1	0	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	1	0	1	Disable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Disable PAUSE receive
1	1	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive

30.11.7.7 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 30-84 describes the definition for the ANLPBPA register.

	0	1	2	3	4		6	7	8	9	10	11				15
R	Next Page	Ack	Remote Fault		0	0	0	Pause		Half Duplex	Full Duplex	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x05															

Figure 30-84. AN Link Partner Base Page Ability Register Definition

Table 30-97 describes the fields of the ANLPBPA register.

Table 30-97. ANLPBPA Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Next Page</td><td>Next page. This bit is read-only. The link partner sets or clears this bit.
0 Link partner has no subsequent next pages or is not capable of receiving next pages.
1 Link partner either requesting next page transmission or indicating the capability to receive next pages.</td></tr>
<tr><td>1</td><td>Ack</td><td>Acknowledge. Ignore on read. This bit is read-only</td></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The link partner’s remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>Encoding of the link partner’s PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult the IEEE 802.3 specification. This bit is read-only
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability. This bit is read-only.
0 Link partner is not capable of half-duplex mode.
1 Link partner is capable of half-duplex mode.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability. This bit is read-only.
0 Link partner is not capable of full-duplex mode.
1 Link partner is capable of full-duplex mode.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

30.11.7.8 AN Expansion Register (ANEX)

Figure 30-85 describes the definition for the ANEX register.

	0												12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	NP Able	Page Rx'd	0
W																
Reset	0000_0000_0000_0110															
Offset	0x06															

Figure 30-85. AN Expansion Register Definition

Table 30-98 describes the fields of the ANEX register.

Table 30-98. ANEX Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13	NP Able	Next page able. This bit is read-only and returns 1 on read. While read as set, indicates local device supports next page function.
14	Page Rx'd	Page received. This bit is read-only. The bit clears on a read to the register. 0 Normal operation. 1 A new page was received and stored in the applicable AN link partner ability or AN next page register. This bit latches high in order for software to detect while polling.
15	—	Reserved, should be cleared.

30.11.7.9 AN Next Page Transmit Register (ANNPT)

Figure 30-86 describes the definition for the ANNPT register.

	0	1	2	3	4	5										15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message / Un-formatted Code Field										
W																
Reset	0000_0000_0000_0000															
Offset	0x07															

Figure 30-86. AN Next Page Transmit Register Definition

Table 30-99 describes the fields of the ANNPT register.

Table 30-99. ANNPT Field Descriptions

Bits	Name	Description
0	Next Page	Next page indication. [Reference MII bit 7.15 in IEEE 802.3, 2000 Edition Clause 28.2.4] 0 Last page. 1 Additional next pages to follow.
1	Ack	Write 0, ignore on read. [Reference MII bit 7.14] This bit is read-only.
2	Msg Page	Message page. [Reference MII bit 7.13] 0 Unformatted page. 1 Message page.
3	Ack2	Acknowledge 2. Used by the next page function to indicate that the device has the ability to comply with the message. [Reference MII bit 7.12] 0 The local device cannot comply with message. 1 The local device complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. [Reference MII bit 7.11] This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was set. 1 Toggle bit of the previously-exchanged link code word was clear.
5–15	Message / Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value. [Reference MII field 7.10:0]

30.11.7.10 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 30-87 describes the definition for the ANLPANP register.

	0	1	2	3	4	5										15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message / Un-formatted Code Field										
W																
Reset	0000_0000_0000_0000															
Offset	0x08															

Figure 30-87. AN Link Partner Ability Next Page Register Definition

Table 30-100 describes the fields of the ANLPANP register.

Table 30-100. ANLPANP Field Descriptions

Bits	Name	Description
0‘	Next Page	Next page. The link partner sets and clears this bit. 0 Last page from link partner. 1 Additional next pages to follow.
1	Ack	Ignore on read. [Reference MII bit 8.14] This bit is read-only.
2	Msg Page	Message page. 0 unformatted page. 1 message page.
3	Ack2	Acknowledge 2. Indicates the link partner’s ability to comply with the message. 0 Link partner cannot comply with message. 1 Link partner complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was set. 1 Toggle bit of the previously-exchanged link code word was clear.
5–15	Message / Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value.

30.11.7.11 Extended Status Register (EXST)

Figure 30-88 describes the definition for the EXST register.

	0	1	2	3	4											15
R	1000X Full	1000X Half	1000T Full	1000T Half	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	1010_0000_0000_0000															
Offset	0x0F															

Figure 30-88. Extended Status Register Definition

Table 30-101 describes the fields of the EXST register.

Table 30-101. EXST Field Descriptions

Bits	Name	Description
0	1000X Full	1000X full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X full-duplex mode. 1 PHY can operate in 1000BASE-X full-duplex mode.
1	1000X Half	1000X half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X half-duplex mode. 1 PHY can operate in 1000BASE-X half-duplex mode.
2	1000T Full	1000T full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T full-duplex mode. 1 PHY can operate in 1000BASE-T full-duplex mode.
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T half-duplex mode. 1 PHY can operate in 1000BASE-T half-duplex mode.
4–15	—	Reserved

30.11.7.12 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. In may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns. Figure 30-89 describes the definition for the JD register.

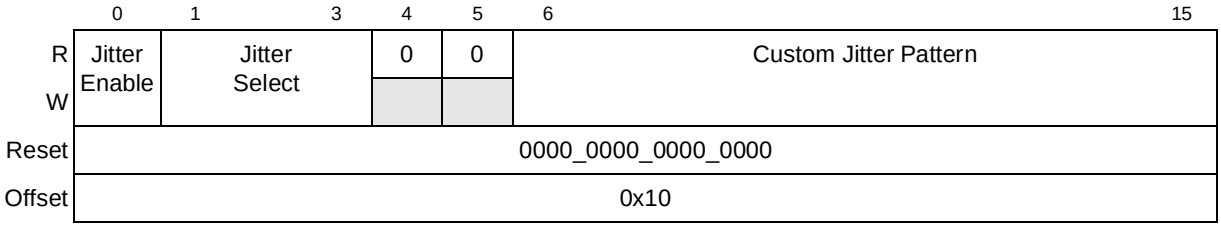


Figure 30-89. Jitter Diagnostics Register Definition

Table 30-102 describes the fields of the JD register.

Table 30-102. JD Field Description

Bits	Name	Description
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation. 1 Enable the TBI to transmit the jitter test patterns defined in IEEE 802.3z 36A.
1–3	Jitter Select	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 00.

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (+/- D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (- K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0
Reserved	1	1	1

Bits	Name	Description
4–5	—	Reserved
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0x000.

30.11.7.13 TBI Control Register (TBICON)

Figure 30-90 describes the definition for the TBICON register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Soft_Reset	0	Disable Rx Dis	Disable Tx Dis	0	0	0	AN Sense	0	0	Clock Select	MII Mode	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x11															

Figure 30-90. TBI Control Register Definition

Table 30-103 describes the fields of the TBICON register.

Table 30-103. TBICON Field Description

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in theTBI.
1	—	Reserved. (Ignore on read)
2	Disable Rx Dis	Disable receive disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the receive direction.
3	Disable Tx Dis	Disable transmit disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the transmit direction.
4–6	—	Reserved
7	AN Sense	Auto-negotiation sense enable. This bit is cleared by default. 0 IEEE 802.3z Clause 37 behavior is desired, which results in the link not completing. 1 Allow the auto-negotiation function to sense either a Gigabit MAC in auto-negotiation bypass mode or an older Gigabit MAC without auto-negotiation capability. If sensed, auto-negotiation complete becomes true; however, the page received is low, indicating no page was exchanged. Management can then act accordingly.
8–9	—	Reserved
10	Clock Select	Clock select. This bit is cleared by default. 0 Allow the TBI to accept dual split-phase 62.5 MHz receive clocks. 1 Configure the TBI to accept a 125 MHz receive clock from the SERDES/PHY. The 125 MHz clock must be physically connected to ‘PMA receive clock 0’.
11	MII Mode	This bit describes the configuration mode of the TBI. The user reads a 1 while the TBI is configured in GMII/MII mode (connected to a GMII/MII PHY) and a 0 while configured in TBI mode (connected to a 1000BASE-X SERDES). Its value is the inverse of ECNTRL[TBIM]. 0 TBI mode. 1 GMII mode.
12–13	—	Reserved
14–15	—	Reserved

30.12 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 30-104 presents the various parameters that impact the multiuser RAM usage.

Table 30-104. Tx and Rx Multiuser RAM Parameters

Parameter	Description	Range	Example for Gigabit Ethernet	Example for Fast Ethernet
TxT	Number of Tx threads	1, 2, 4, 6	4	1
TxQ	Number of Tx queues	1 to 8	4	4
RxT	Number of Rx threads	1, 2, 4, 6	4	1
TxVF	Tx Virtual FIFO size [bytes]	408 or more	2048	272
RxQ	Number of Rx queues	1 to 8	4	4
RxP	Number of PCDs	0 to 3	2	0
RxL	Rx lookup table size [bytes]	32, 48, 80	80	0
RxVF	Rx Virtual FIFO size [bytes]	272 or more	3072	272

Table 30-105 presents the Tx Parameter RAM memory consumption, according to the setting of the appropriate parameters.

Table 30-105. 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	416	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:				1216		744

Table 30-106 presents the Tx Parameter RAM memory consumption, according to the setting of the appropriate parameters.

Table 30-106. 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
ThreadQ	40	RxT	4	160	1	40
Ethernet Statistics counters	92	1	1	92	1	92

Table 30-106. Rx Parameter RAM Usage (continued)

Data structure	Size [bytes]	Multiply by	Gigabit Ethernet example		Fast Ethernet example	
			Value	Total	Value	Total
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	128	RxT	4	512	1	128
Total Rx Parameter RAM usage:				1340		740

Table 30-107 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 30-104.

Table 30-107. 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 30-104 and follow the example in the tables, Table 30-105 to Table 30-107.

30.13 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.

30.13.1 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.

30.13.1.1 Ethernet no LLC Header Format

Table 30-108. Ethernet no LLC Header Format

MAC Destination Address	
MAC Destination Address (cont)	MAC Source Address
MAC Source Address (cont)	
EType (>1536)	

30.13.1.2 802.3, 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 30-109. 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		

30.13.1.3 802.3, 802.2 SNAP LLC

Table 30-110. 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

30.13.2 Ethernet/802.3 with VLAN

30.13.2.1 VTagged Ethernet Encapsulation

Table 30-111. 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		

30.13.2.2 VTagged 802.3/802.2 SNAP Encapsulation

Table 30-112. 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		
Control = 0x03	OUI 00-00-00		
EType	L3 Header		

30.13.3 PPPoE+PPP (RFC2516, RFC1661)

Table 30-113. 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.

30.13.4 I Pv4 (RFC791)

Table 30-114. 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)					

30.13.5 UDP (RFC768)

Table 30-115. UDP (RFC768)

Portsrc	Portdst
Length	Checksum

30.13.6 TCP (RFC793)

Table 30-116. TCP (RFC793)

Portsrc			Portdst
sequence number			
ack number			
data offset	reserved	flags	window
checksum			urgent pointer
options			padding

30.14 Exact Match Tags Memory Organization

This appendix describes the memory organization of the Exact Match Tags in Hash Mode Table Lookup. See Section 30.6.2.6, “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 30-91. Eight Bytes Exact Match Tag (4 Sets)

Table 30-117. 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 30-92. Sixteen Bytes Exact Match Tag (4 Sets)

Table 30-118. Sixteen 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.

30.15 Traffic Shaper Programming Considerations

Definitions

- AvBR - Average Bit Rate. The average bit rate desired at the port during transmission
- MBL - Maximum Burst Length. The longest 1Gbps 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 = 128KB

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

30.15.1 Programming Examples for the Traffic Shaping Characteristics of the Ethernet Tx Distributor

Table 30-119 and Table 30-120 show the user-defined variables.

Table 30-119. 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 30-120. 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)

30.15.1.1 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

30.15.1.2 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

30.15.1.3 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)

30.15.1.4 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 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **NorTSRByteTime** = 0xA4 (from Section 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **MBLInterval** - 0x000051EC (from Section 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)

30.15.1.5 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 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **NorTSRByteTime** = 0xA4 (from Section 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)
- Set **MBLInterval** - 0x000051EC (from Section 30.15.1.3, “Example: Rate Limit All Queues with Maximum Burst Length”)

30.15.1.6 Example: Weighted Fair Queuing without Rate Limiting

Rate limiting is disabled. Weighted Fair Queuing is enabled on all queues.

Table 30-121. 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 30-122. 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 30-122

30.15.1.7 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 30-123. 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 30-123

30.15.1.8 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 30-124. 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 30-124. 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	1/10		
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 31
QUICC Engine IEEE1588 Assist

31.1 Introduction

The objective of the IEEE1588 standard is to specify a protocol to synchronize independent clocks running on separate nodes of a distributed measurement and control system to a high degree of accuracy and precision. The clocks communicate with each other over a communication network. The protocol generates a master-slave relationship among the clocks in the system. Within a given subnet of a network, there will be a single master clock. All clocks ultimately derive their time from a clock known as the grandmaster clock.

The protocol will enable heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize. The protocol will support systemwide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications will be enhanced by having an accurate systemwide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Existing protocols for clock synchronization are not optimal for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with millisecond synchronization requirements.

Although IEEE1588 allows software-only implementations, hardware-assisted implementations deliver more precise clock synchronization.

The simplest IEEE1588 implementations include ordinary applications at the top of the network protocol stack and generate timestamps at the application level. Typically, this implementation incurs the largest protocol stack delay fluctuation, thus yielding the least accuracy as the largest amount of error is introduced into the timestamp.

Hardware-assisted methods achieve the greatest accuracy due to the fact that they generate timestamps as close to the wire as possible, at the physical layer. Network-protocol-delay fluctuations for these implementations typically range from nanoseconds to sub-microseconds.

In the QUICC Engine block, the IEEE1588 implementation is a Hardware-assisted method. The hardware assist includes a time stamp unit to recognize PTP frames and transfer relevant timestamps, and a real time clock with high resolution.

31.2 QUICC Engine IEEE1588 Block Diagram

Figure 31-1 shows the TSU (Time Stamp Unit) and the IEEE1588 RT (Realtime Clock) modules integrated into the QUICC Engine.

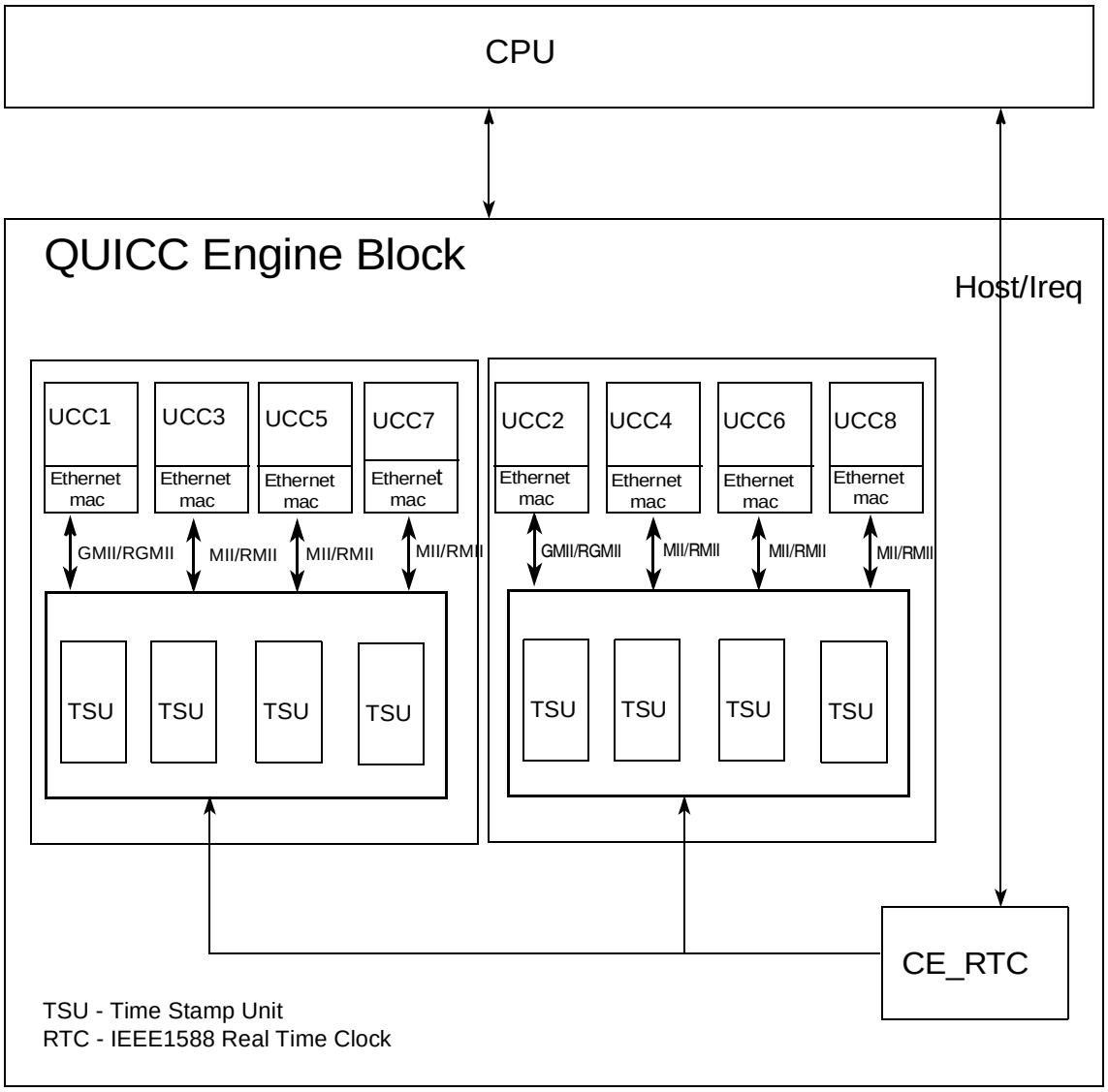


Figure 31-1. QUICC Engine IEEE1588 Block Diagram

31.3 Time Stamp Unit Key Features:

Key Features of the Time Stamp Unit are:

- Supports Ethernet systems
- Support IEEE1588 timestamping on all UCC ports
- Recognition of PTP frames on receive side
  - — Sync type
  - — Delay_req type
  - — Delay_resp type
  - — Follow_Up type
  - — Management type

- Support PTP frame with VLAN tag (single VLAN)
- Supports MII/RMII/GMII/RGMII Physical I/F
- Support Ethernet frame monitoring on full and half duplex modes
- Supports slave and master mode
- Support single- or multi- IEEE1588 masters
- Support out-of-band and in-band modes of operation.
  - in-band (timestamp is attached to each Ethernet frame buffer)
  - out-of-band
- Timestamp capture close to the physical line (symmetric for transmitter and receiver)
- Supports user configured values for PTP header fields.
- Support user configured value for fields offset.
- Support timestamp overrun report for Tx and RX
- Supports interrupt notification due to following:
  - Rx PTP frame detection (only delay_req or sync)
  - Tx PTP frame transmission which was marked by the Software as a PTP frame
  - Rx and Tx Timestamp overrun error

31.4 RTC Key Features (IEEE1588 Real Time Clock)

- Support single IEEE1588 Real Time Clock
- Support timer frequency compensation.
- Support timer offset update.
- Support nanosecond resolution time stamps.
- Support configured source of clock
  - External oscillator
  - Transmitter serial clock up to 125MHz.
  - Half System clock
  - BRG divided clock
- Time stamp capture on two general purpose external triggers through new GPIO connection
  - Maskable interrupts on GPIO timestamp trigger
  - Programmable polarity of external trigger (GPIO) edge
- Two 64-bit alarm (future time) registers for future time comparison
  - Maskable interrupts on alarm
- Three programmable timer output pulse period phase aligned with IEEE1588 timer clock
  - Maskable interrupts associated with each pulse
- Separate maskable timer interrupt event register
- Phase aligned adjustable (divide by N) clock output

31.5 IEEE1588 Implementation Assumptions

The Version 1 PTP standard makes several assumptions about the environment in which it operates. The following assumptions must be met to ensure correct operation of the Version 1 protocol:

- The network must support multicast communication.
- It must be possible to prevent multicast messages from propagating beyond a subnet.
- Each clock implementing the protocol must meet the physical performance requirements, including oscillator frequency and time adjustment range.
- A clock's stated properties, including its stratum and identifier, must accurately describe the clock.

The following assumptions must be met to achieve optimal clock synchronization performance:

- Network delay between master and slave on a subnet must be symmetric.
- A clock may contain asymmetric delays in its timestamp mechanism or protocol path. If these asymmetries are not negligible, they must be correctly accounted for.
- Network delay between master and slave on a subnet must be constant.
- Boundary clocks must be used to synchronize across subnets.
- The timestamps used in PTP are generated as close to the physical layer as practical for a given clock implementation. In cases where the most accurate timestamps can be generated only after a message has actually been transmitted, the actual value is communicated in the Follow_Up message from the master clock.
- The computing power of clocks implementing the protocol must be great enough, and the number of clocks per subnet must be small enough, to meet the standard constraints.
- The inherent stability of a clock s oscillator must be adequate.

31.6 Modes of Operation

- In-band - This mode is determined in the TSMR register. The receiver's timestamps are transferred as part of the data payload. The user sets the PTP bit in the TxBD while sending a PTP frame, and gets a PTP indication in the RxBD when a PTP frame has received.
- Out-of-band - This mode is determined in the TSMR register. All the timestamps are transferred through the host interface, using dedicated registers.The host reads the transmitter's timestamps after last indication in the buffer descriptor, and the receiver's timestamps when a PTP bit is set in the RxBD. The user is responsible for setting the PTP bit in the TxBD. In both receiver and transmitter the user can set an interrupt before accessing the timestamps.
- PTP frame parsing options - All of the relevant parsing values and fields can be configured using TSPDR1-4 registers, retaining high flexibility.
- Loopback - The TSU Rx monitored frame is the Tx frame which being transmitted by the same UCC. In order to work in this mode, the user should set the loopback bit in the UCC GUMR mode register (GUMR[DIAG] = 01).
- Pulse per second - The user can configure the fiper registers (TMR_FIPERn) in order to generate the pulse per second outputs.

- Alarm mode - The user can configure a dedicated register as an alarm value. When the IEEE1588 RTC reaches this value, an interrupt is generated to the core, and a dedicated output trigger signal will be asserted.
- Input trigger - The user can synchronously trigger the IEEE1588 RTC in order to capture a timestamp.
- Source clock select - This mode is configured in the Timer Control Register (TMR_CTRL).The user can connect an external clock to the IEEE1588 RTC. In addition, the RTC can work with the system clock or with the Ethernet physical interface clock.

31.7 Memory Map/Register Definition

All IEEE1588 registers are 32-bits wide, are located on 32-bit address boundaries, and should only be accessed as 32-bit quantities.

All addresses used in this chapter are offsets from RTC Base, PTP1 Base and PTP2 Base. Table 31-1 lists the PTP1 Registers.

Table 31-1. PTP1 Registers

Offset	Name	Address	Size	R/W	Destination
Time Stamp Unit Mode Registers					
0x0	PTP1_TSPDR1	0x4880	32 bit	R/W	internal
0x4	PTP1_TSPDR2	0x4884	32 bit	R/W	internal
0x8	PTP1_TSPDR3	0x4888	32 bit	R/W	internal
0xc	PTP1_TSPDR4	0x488C	32 bit	R/W	internal
0x10	PTP1_TSPOV	0x4890	32 bit	R/W	internal
0x14	PTP1_TSMR	0x4894	32 bit	R/W	internal
0x18	PTP1_TMR_PEVENT	0x4898	32 bit	R/W	core
0x1C	PTP1_TMR_PEMASK	0x489C	32 bit	R/W	internal
0x20	TMR_UC1_RXTS_H	0x48A0	32 bit	R	host
0x24	TMR_UC3_RXTS_H	0x48A4	32 bit	R	host
0x28	TMR_UC5_RXTS_H	0x48A8	32 bit	R	host
0x2C	TMR_UC7_RXTS_H	0x48AC	32 bit	R	host
0x30	TMR_UC1_RXTS_L	0x48B0	32 bit	R	host
0x34	TMR_UC3_RXTS_L	0x48B4	32 bit	R	host
0x38	TMR_UC5_RXTS_L	0x48B8	32 bit	R	host
0x3C	TMR_UC7_RXTS_L	0x48BC	32bit	R	host
0x40	TMR_UC1_TXTS_H	0x48C0	32bit	R	host
0x44	TMR_UC3_TXTS_H	0x48C4	32bit	R	host

Table 31-1. PTP1 Registers (continued)

Offset	Name	Address	Size	R/W	Destination
Time Stamp Unit Mode Registers					
0x48	TMR_UC5_TXTS_H	0x48C8	32bit	R	host
0x4C	TMR_UC7_TXTS_H	0x48CC	32bit	R	host
0x50	TMR_UC1_TXTS_L	0x48D0	32bit	R	host
0x54	TMR_UC3_TXTS_L	0x48D4	32bit	R	host
0x58	TMR_UC5_TXTS_L	0x48D8	32bit	R	host
0x5C	TMR_UC7_TXTS_L	0x48DC	32bit	R	host

Table 31-2 lists the PTP2 Registers.

Table 31-2. PTP2 Registers

offset	Name	Address	size	R/W	destination
0x0	PTP2_TSPDR1	0x4900	32bit	R/W	internal
0x4	PTP2_TSPDR2	0x4904	32bit	R/W	internal
0x8	PTP2_TSPDR3	0x4908	32bit	R/W	internal
0xC	PTP2_TSPDR4	0x490C	32bit	R/W	internal
0x10	PTP2_TSPOV	0x4910	32bit	R/W	internal
0x14	PTP2_TSMR	0x4914	32bit	R/W	internal
0x18	PTP2_TMR_PEVENT	0x4918	32bit	R/W	core
0x1C	PTP2_TMR_PEMASK	0x491C	32bit	R/W	internal
0x20	TMR_UC2_RXTS_H	0x4920	32bit	R	host
0x24	TMR_UC4_RXTS_H	0x4924	32bit	R	host
0x28	TMR_UC6_RXTS_H	0x4928	32bit	R	host
0x2C	TMR_UC8_RXTS_H	0x492C	32bit	R	host
0x30	TMR_UC2_RXTS_L	0x4930	32bit	R	host
0x34	TMR_UC4_RXTS_L	0x4934	32bit	R	host
0x38	TMR_UC6_RXTS_L	0x4938	32bit	R	host
0x3C	TMR_UC8_RXTS_L	0x493C	32bit	R	host
0x40	TMR_UC2_TXTS_H	0x4940	32bit	R	host
0x44	TMR_UC4_TXTS_H	0x4944	32bit	R	host
0x48	TMR_UC6_TXTS_H	0x4948	32bit	R	host
0x4C	TMR_UC8_TXTS_H	0x494C	32bit	R	host
0x50	TMR_UC2_TXTS_L	0x4950	32bit	R	host

Table 31-2. PTP2 Registers (continued)

offset	Name	Address	size	R/W	destination
0x54	TMR_UC4_TXTS_L	0x4954	32bit	R	host
0x58	TMR_UC6_TXTS_L	0x4958	32bit	R	host
0x5C	TMR_UC8_TXTS_L	0x495C	32bit	R	host

Table 31-3 lists the RTC Registers.

Table 31-3. RTC Registers

offset	Name	Address	size	R/W	destination
IEEE1588 Timer Mode Registers					
0x0	TMR_CTRL	0x4800	32bit	R/W	internal
0x4	TMR_TEVENT	0x4804	32bit	R/W	core
0x8	TMR_TEMASK	0x4808	32bit	R/W	internal
0xC	TMR_CNT_L	0x480C	32bit	R/W	host
0x10	TMR_CNT_H	0x4810	32bit	R/W	host
0x14	TMR_ADD	0x4814	32bit	R/W	internal
0x18	TMR_ACC	0x4818	32bit	R	internal
0x1C	TMR_PRSC	0x481C	32bit	R/W	internal
0x20	TMROFF_L	0x4820	32bit	R/W	internal
0x24	TMROFF_H	0x4824	32bit	R/W	internal
0x28	TMR_ALARM1_L	0x4828	32bit	R/W	internal
0x2C	TMR_ALARM1_H	0x482C	32bit	R/W	internal
0x30	TMR_ALARM2_L	0x4830	32bit	R/W	internal
0x34	TMR_ALARM2_H	0x4834	32bit	R/W	internal
0x38	TMR_FIPER1	0x4838	32bit	R/W	internal
0x3C	TMR_FIPER2	0x483C	32bit	R/W	internal
0x40	TMR_FIPER3	0x4840	32bit	R/W	internal
0x44	TMR_ETTS1_L	0x4844	32bit	R	host
0x48	TMR_ETTS1_H	0x4848	32bit	R	host
0x4C	TMR_ETTS2_L	0x484C	32bit	R	host
0x50	TMR_ETTS2_H	0x4850	32bit	R	host

31.8 Time Stamp Unit Mode Registers

31.8.1 Time Stamp Unit Parsing Definitions Register 1 (PTPn_TSPDR1)

The Time Stamp Unit Parsing Definition Register (PTPn_TSPDR1) is shown in Figure 31-2.

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame. The PTPn_TSPDR1 register is used to determine the values of the Ethernet type field (12th–13th octets) and the IP type(23rd octet). The default values are set to 0x0800 and 0x11.

The user can modify the values for the parsing.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ETT															
Reset	0x0800															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	IPT								—							
Reset	0x11								—							
R/W	R/W															
Addr	4880/4900															

Figure 31-2. PTPn_TSPDR1 Register

Table 31-4 describes the PTPn_TSPDR1 fields.

Table 31-4. PTPn_TSPDR1 Field Descriptions

Bits	Name	Description
0–15	ETT	Ethernet frame type. IP frame by default. This field indicates if the observed frame is a IP frame(0x800).
16–23	IPT	IP frame type. UDP frame by default. This field indicates if observed frame is UDP frame(0x11).
24–31	—	Reserved

31.8.2 Time Stamp Unit Parsing Definitions Register 2(PTPn_TSPDR2)

The Time Stamp Unit Parsing Definition Register 2(PTPn_TSPDR2) is shown in Figure 31-3.

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame. The PTPn_TSPDR2 register is used to determine the values of the destination port number field (36th-37th octets). For PTP sync and delay_req messages the port number should 0x13f (319). For the rest, 0x140 (320).

The user can modify the values for the parsing.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DPNGE															
Reset	0x0140															
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DPNEV															
Reset	0x013f															
R/W	R/W															
Addr	4884/4904															

Figure 31-3. PTPn_TSPDR2 Register

Table 31-5 describes the PTPn_TSPDR2 fields.

Table 31-5. PTPn_TSPDR2 Field Descriptions

Bits	Name	Description
0–15	DPNGE	Destination port number of a PTP general messages - Follow_up, Delay_resp and management. (0x140).
16–31	DPNEV	Destination port number of a PTP event messages - Sync and delay_req(0x13f).

31.8.3 Time Stamp Unit Parsing Definitions Register 3 (PTPn_TSPDR3)

The Time Stamp Unit Parsing Definition Register 3 (PTPn_TSPDR3) is shown in Figure 31-4.

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame.

The specific type of frame is derived from the control field, 0x0 for sync message, 0x1 for Delay_req message, 0x2 for follow up, 0x3 for delay_resp.

The user can modify the values for the parsing.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SYCTL								DRCTL							
Reset	0x00								0x01							
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DRPCTL								FUCTL							
Reset	0x03								0x02							
R/W	R/W															
Addr	4888/4908															

Figure 31-4. PTPn_TSPDR3 Register

Table 31-6 describes the PTPn_TSPDR3 fields.

Table 31-6. PTPn_TSPDR3 Field Descriptions

Bits	Name	Description
0–7	SYCTL	Control field value represents sync message(0x0).
7–15	DRCTL	Control field value represents Delay_Req message(0x1).
16–23	DRPCTL	Control field value represents delay_resp message(0x2).
24–31	FUCTL	Control field value represents follow_up message(0x3).

31.8.4 Time Stamp Unit Parsing Definitions Register 4 (PTPn_TSPDR4)

The Time Stamp Unit Parsing Definition Register 4 (PTPn_TSPDR4) is shown in Figure 31-5.

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame.

The specific type of frame is derived from the control field, 0x4 for management message.

In addition, PTPn_TSPDR4 register provides a configured value to detect a VLAN field. This value can be detected during bytes 12-13 and cause a shifting of 4 bytes for all other parsing offsets. In case of VLAN frame, the user should configure the offsets exactly as for non VLAN frame.

The user can modify the values for the parsing.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MACTL								—							
Reset	0x04								0x00							
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	VLAN															
Reset	0x8100															
R/W	R/W															
Addr	488c/490c															

Figure 31-5. PTPn_TSPDR4 Register

Table 31-7 describes the PTPn_TSPDR4 fields.

Table 31-7. PTPn_TSPDR4 Field Descriptions

Bits	Name	Description
0–7	MACTL	Control field value represents management message.
8–15	—	Reserved
16–31	VLAN	VLAN tag (type=0x8100)

31.8.5 Time Stamp Unit Parsing Offset Values (PTPn_TSPOV)

The Time Stamp Unit Parsing Offset Values (PTPn_TSPOV) is shown in Figure 31-6.

In order to detect a PTP frame, the TSU should parse several fields in the Ethernet frame. TSU offset values register (PTPn_TSPOV) is used to determine the offset of each field in the Ethernet frame. The user can change the fields offset in case the structure of the frame has been changed or manipulated.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	ETTOF								IPTOF							
Reset	12								23							
R/W																
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	UDOF								PTOF							
Reset	36								74							
R/W	R/W															
Addr	4890/4910															

Figure 31-6. PTPn_TSPOV Register

Table 31-8 describes the PTPn_TSPOV fields.

Table 31-8. PTPn_TSPOV Field Descriptions

Bits	Name	Description
0–7	ETTOF	Ethernet type offset(13)
8–15	IPTOF	IP type offset(23)
16–23	UDOF	UDP type offset(37)
24–31	PTOF	PTP type offset(74)

31.8.6 Time Stamp Unit Mode Register (PTPn_TSMR)

The Time Stamp Unit Mode Register (PTPn_TSMR) is shown in Figure 31-7.

The PTPn_TSMR register is used to determine the modes of operation of the TSU.

The TSU can operate in both in-band or out-of-band methods.

In addition, PTPn_TSMR EN bits enable the TSU operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—												OPMODE1	OPMODE2	OPMODE3	OPMODE4
Reset													0	0	0	0
R/W	R/W															
Addr																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—												EN1	EN2	EN3	EN4
Reset													0	0	0	0
R/W	R/W															
Addr	4894/4914															

Figure 31-7. PTPn_TSMR Register

Table 31-9 describes the PTPn_TSMR fields.

Table 31-9. PTPn_TSMR Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	OPMODE1	TSU mode of operation. UCC1(PTP1), UCC2(PTP2) 0 - out-of-band 1 - in-band Note: if OPMODE is set the Ethernet receive soft preamble mode should be set as well (MACCFG2[SRP])
13	OPMODE2	TSU mode of operation. UCC3(PTP1),UCC4(PTP2) 0 - out-of-band 1 - in-band Note: if OPMODE is set the Ethernet receive soft preamble mode should be set as well (MACCFG2[SRP])
14	OPMODE3	TSU mode of operation. UCC5(PTP1),UCC6(PTP2) 0 - out-of-band 1 - in-band Note: if OPMODE is set the Ethernet receive soft preamble mode should be set as well (MACCFG2[SRP])
15	OPMODE4	TSU mode of operation. UCC7(PTP1),UCC8(PTP2) 0 - out-of-band 1 - in-band Note: if OPMODE is set the Ethernet receive soft preamble mode should be set as well (MACCFG2[SRP])
16–27	—	Reserved
28	EN1	0 - Time Stamp Unit UCC1(PTP1)/UCC2(PTP2) Disabled 1- Time Stamp UCC1(PTP1)/UCC2(PTP2) Unit Enabled.

Table 31-9. PTPn_TSMR Field Descriptions (continued)

Bits	Name	Description
29	EN2	0 - Time Stamp Unit UCC3(PTP1)/UCC4(PTP2) Disabled 1- Time Stamp Unit UCC3(PTP1)/UCC4(PTP2) Enabled.
30	EN3	0 - Time Stamp Unit UCC5(PTP1)/UCC6(PTP2) Disabled 1- Time Stamp Unit UCC5(PTP1)/UCC6(PTP2) Enabled.
31	EN4	0 - Time Stamp Unit UCC7(PTP1)/UCC8(PTP2) Disabled 1- Time Stamp Unit UCC7(PTP1)/UCC8(PTP2) Enabled.

31.8.7 Timer PTP Event Register (PTPn_TMR_PEVENT)/ Timer PTP Mask Register (PTPn_TMR_PEMASK)

The Time Stamp Unit Event Register (PTPn_TMR_PEVENT) and Time Stamp Mask Register are shown in Figure 31-8.

The PTPn_TMR_PEVENT is used as the TSU event register. The PTPn_TMR_PEVENT reports events recognized on the Ethernet channel and generates interrupts. After event recognition, the TSU sets the corresponding PTPn_TMR_PEVENT bit. The PTPn_TMR_PEVENT bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the CPU clears the internal interrupt request.

Interrupts generated by the TSU event register (PTPn_TMR_PEVENT) can be masked in the TSU mask register (PTPn_TMR_PEMASK), which has the same bit format as PTPn_TMR_PEVENT. If a PTPn_TMR_PEMASK bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

	0			3	4	5										15
Field	OVR1	OVT1	SYRE1	DRQRE1	TXE1	OVR2	OVT2	SYRE2	DRQRE2	TXE2	OVR3	OVT3	SYRE3	DRQRE3	TXE3	OVR4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															
Addr																
	16			19												31
Field	OVT4	SYRE4	DRQRE4	TXE4	—											
Reset	0	0	0	0												
R/W	R/W															
Addr	4898/4918 489c/491c															

Figure 31-8. Event Register (TMR_PEVENT) / Mask Register (TMR_PEMASK)

Table 31-10 describes the TMR_PEVENT/TMR_PEMASK fields.

Table 31-10. TMR_PEVENT/TMR_PEMASK Field Descriptions

Bits	Name	Description
0	OVR1	Overrun occurred in receiver timestamp register. UCC1(PTP1), UCC2(PTP2)
1	OVT1	Overrun occurred in transmitter timestamp register. UCC1(PTP1), UCC2(PTP2)
2	SYRE1	Sync frame has been received. UCC1(PTP1), UCC2(PTP2)
3	DRQRE1	Delay_req frame has been received. UCC1(PTP1), UCC2(PTP2)
4	TXE1	PTP frame has been transmitted. UCC1(PTP1), UCC2(PTP2)
5	OVR2	Overrun occurred in receiver timestamp register. UCC3(PTP1), UCC4(PTP2)
6	OVT2	Overrun occurred in transmitter timestamp register. UCC3(PTP1), UCC4(PTP2)
7	SYRE2	Sync frame has been received. UCC3(PTP1), UCC4(PTP2)
8	DRQRE2	Delay_req frame has been received. UCC3(PTP1), UCC4(PTP2)
9	TXE2	PTP frame has been transmitted UCC3(PTP1), UCC4(PTP2)
10	OVR3	Overrun occurred in receiver timestamp register. UCC5(PTP1), UCC6(PTP2)
11	OVT3	Overrun occurred in transmitter timestamp register. UCC5(PTP1), UCC6(PTP2)
12	SYRE3	Sync frame has been received. UCC5(PTP1), UCC6(PTP2)
13	DRQRE3	Delay_req frame has been received. UCC5(PTP1), UCC6(PTP2)
14	TXE3	PTP frame has been transmitted. UCC5(PTP1), UCC6(PTP2)
15	OVR4	Overrun occurred in receiver timestamp register. UCC7(PTP1), UCC8(PTP2)
16	OVT4	Overrun occurred in transmitter timestamp register. UCC7(PTP1), UCC8(PTP2)
17	SYRE4	Sync frame has been received. UCC7(PTP1), UCC8(PTP2)
18	DRQRE4	Delay_req frame has been received. UCC7(PTP1), UCC8(PTP2)
19	TXE4	PTP frame has been transmitted UCC7(PTP1), UCC8(PTP2)
20–31	—	RESERVED

31.8.8 Time Stamp Unit Receiver Time High (TMR_UCn_RXTS_H)/Time Stamp Unit Receiver Time Low (TMR_UCn_RXTS_L)/Time Stamp Unit Transmitter Time High (TMR_UCn_TXTS_H)/Time Stamp Unit Transmitter Time Low (TMR_UCn_TXTS_L)

The TMR_UCn_TXTS_L/TMR_UCn_TXTS_H and TMR_UCn_RXTS_L/TMR_UCn_RXTS_H registers are shown in Figure 31-9.

The TMR_UCn_TXTS_L/TMR_UCn_TXTS_H registers are used to capture the time stamp of a recognized PTP frame on the transmitter lines. The TSU (Time Stamp Unit) samples the IEEE1588 RTC (Real Time Clock) 64-bit counter, immediately after SFD (Start Frame Delimiter) indication. The sampled

value becomes valid only after one of the frame events. In this case, the CPU reads the value in order to calculate the synchronizations commands.

TMR_UCn_RXTS_L/TMR_UCn_RXTS_H has the same purpose on the receiver lines.

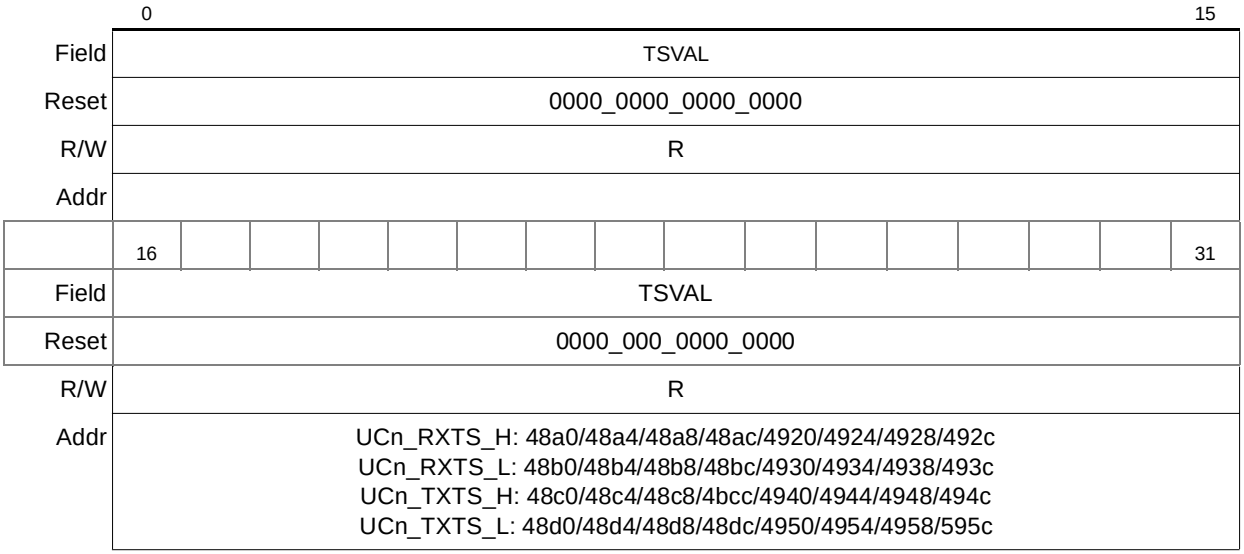


Figure 31-9. TMR_UCn_RXTS_H, TMR_UCn_RXTS_L, TMR_UCn_TXTS_H, TMR_UCn_TXTS_L Registers

Table 31-11 describes the fields.

Table 31-11. TMR_UCn_TXTS_L, TMR_UCn_RXTS_L, TMR_UCn_TXTS_H, TMR_UCn_RXTS_H Field Descriptions

Bits	Name	Description
0–31	TSVAL	Time stamp value of the PTP packet's start of frame detection.

31.9 IEEE1588 Timer Mode Registers

31.9.1 Timer Control Register (TMR_CTRL)

The Timer Control register defines control fields relevant to the IEEE1588 timer. TMR_CTRL is a register writable by the user to reset, configure, and initialize the QUICC Engine IEEE1588 timer clock. Only one of these registers is active for the entire QUICC Engine. Figure 31-10 describes the definition for the TMR_CTRL register.

	0	1	2	3	4	6	15
Field	ALM1P	ALM2P		FS		TCLK_PERIOD	
Reset	0	0		o		000_0000_0000	
R/W	R/W						
Addr							

	16			19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—						ETEP2	ETEP1	COPH	CIPH	TSMR	DBG	BYP	TE	CKSEL	
Reset	0000_00						0	0	0	0	0	0	0	0	00	
R/W	R/W															
Addr	0x4800															

Figure 31-10. TMR_CTRL Register

Table 31-12 describes the TMR_CNTL fields.

Table 31-12. TMR_CTRL Field Descriptions

Bits	Name	Description
0	ALM1P	Alarm1 output polarity 0 active high output 1 active low output
1	ALM2P	Alarm2 output polarity 0 active high output 1 active low output
2	—	Reserved
3	FS	Fiper start indication 0 fiper countdown start is enabled through timer enable 1 fiper countdown start is enabled through timer enable and alarm indication. Note: Once set, FS will remain at 1 until cleared by software.
4–5	—	Reserved
6–15	TCLK_PERIOD	IEEE1588 timer reference clock period. The timer clock counter will increment by TCLK_PERIOD every time the accumulator register overflows. This clock period must be larger than the clock period of the timer reference clock. For applications where user does not want the clock period to be added, they can program this field to 1 to count the clock ticks. This field defaulted to 1 to count overflow ticks.
16–19	—	Reserved
22	ETEP2	External trigger 1 edge polarity 0 time stamp on the rising edge of the external trigger 1 time stamp on the falling edge of the external trigger
23	ETEP1	External trigger 2 edge polarity 0 time stamp on the rising edge of the external trigger 1 time stamp on the falling edge of the external trigger

Table 31-12. TMR_CTRL Field Descriptions (continued)

Bits	Name	Description
24	COPH	Generated clock output phase. 0 non-inverted divided clock is output 1 inverted divided clock is output
25	CIPH	External oscillator input clock phase. 0 non-inverted frequency tuned timer input clock 1 inverted frequency tuned timer input clock
26	TMSR	Timer soft reset. When enabled, it resets all the timer registers and state machines. 0 normal operation 1 places entire timer in reset except control and config registers
27	DBG	Enable debug mode. When set, it allows write to the read only timer registers.
28	BYP	Bypass drift compensated clock 0 64-bit clock counter is incremented on the accumulator overflow 1 64-bit clock counter is directly driven from the external oscillator, ignoring accumulator overflow
29	TE	IEEE1588 timer enable. If not enabled, all the timer registers and state machines are disabled. 0 timer not enabled 1 timer enabled and resumes normal operation
30–31	CKSEL	IEEE1588 Timer reference clock source select. 00 External high precision timer reference clock 01 system clock 10 transmit clock 11 RTC clock oscillator (NA for QUICC Engine)

31.9.2 Timer Event Register (TMR_TEVENT)/Timer Event Mask Register (TMR_TEMASK)

The IEEE1588 timer implementation requires several interrupt event signals. The QUICC Engine needs to add a new interrupt output line for the timer independent of the existing Tx, Rx and Error interrupts. The IEEE1588 timer interrupt does not require any interrupt coalescing. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the event mask register (EMASK), the event also causes a hardware interrupt at the PIC. A bit in the timer event register is cleared by writing a 1 to that bit position.

Figure 31-11shows the TMR_EVENT register.

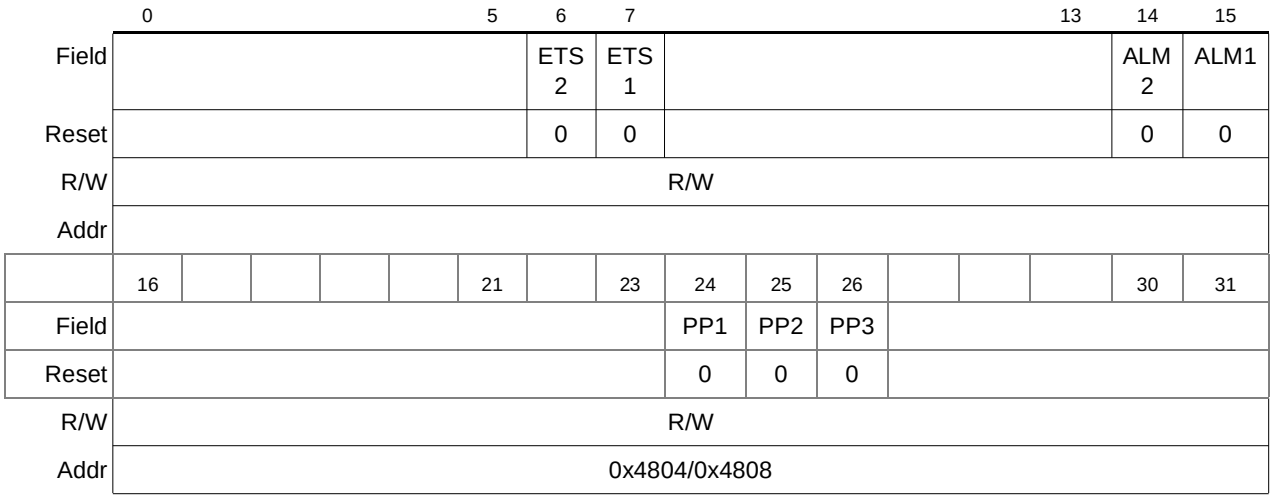


Figure 31-11. Event Register (TMR_TEVENT) / Mask Register (TMR_TEMASK)

Table 31-13 describes the fields.

Table 31-13. TMR_TEVENT/TMR_TEMASK Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	ETS2	External trigger 2 timestamp sampled 0 external trigger timestamp not sampled 1 external trigger timestamp sampled
7	ETS1	External trigger 1 timestamp sampled 0 external trigger timestamp not sampled 1 external trigger timestamp sampled
8–13	—	Reserved
14	ALM2	Current time equaled alarm time register 1 0 alarm time has not be reached yet 1 alarm time has been reached
15	ALM1	Current time equaled alarm time register 2 0 alarm time has not be reached yet 1 alarm time has been reached
16–23	—	Reserved
24	PP1	Indicates that a periodic pulse has been generated based on FIPER1 register. 0 periodic pulse not generated 1 periodic pulse generated
25	PP2	Indicates that a periodic pulse has been generated based on FIPER2 register. 0 periodic pulse not generated 1 periodic pulse generated

Table 31-13. TMR_TEVENT/TMR_TEMASK Field Descriptions (continued)

Bits	Name	Description
26	PP3	Indicates that a periodic pulse has been generated based on FIPER2 register. 0 periodic pulse not generated 1 periodic pulse generated
27–31	—	Reserved

31.9.3 Timer Counter Register (TMR_CNT_L/TMR_CNT_H)

The TMR_CNT_L/ TMR_CNT_H registers are shown in Figure 31-12.

The timer register (TMR_CNT_L/TMR_CNT_H) represents accurate time in terms clock ticks or in nano-seconds. This is a read/write register. Writes to these registers will override the previous time.

	0															15
Field	TMR_CNT															
Reset	0000_0000_0000_0000															
R/W	W/R															
Addr																
	16													29	30	31
Field	TMR_CNT															
Reset	0000_0000_0000_0000															
R/W	W/R															
Addr	0x480c/0x4810															

Figure 31-12. TMR_CNT_L/TMR_CNT_H Register

Table 31-14 describes the fields.

Table 31-14. TMR_CNT_L/TMR_CNT_H Field Descriptions

Bits	Name	Description
0–31	TMR_CNT	H/L Value of the current time counter. Current time is calculated by adding TMROFF_H/L with the TMR_CNT_H/L counter. This register can be written through the register writes. Writes to the TMR_CNT_L register copies the written value into the shadow TMR_CNT_L register. Writes to the TMR_CNT_H register copies the values written into the shadow TMR_CNT_H register. Contents of the shadow registers are copied into the TMR_CNT_L and TMR_CNT_H registers following a write into the TMR_CNT_H register. Writes to these registers have precedence over the timer increment. The user must write to TMR_CNT_L register first. Reads from the TMR_CNT_L register copies the entire 64-bit clock time of the read enable into the TMR_CNT_H/L shadow registers. Read instruction from the TMR_CNT_H register reads the value stored in the TMR_CNT_H shadow register. The user must read the TMR_CNT_L register first to get correct 64-bit TMR_CNT_H/L counter values.

31.9.4 Timer Addend register (TMR_ADD)

The Timer drift compensation Addend register (TMR_ADD) is used to hold the timer frequency compensation value (FreqCompensationValue). The nominal frequency of the clock counter is determined by the FreqDivRatio and the clock frequency (FreqClock). This register is programmed with 2^{32}/FreqDivRatio. Frequency division ratio (FreqDivRatio) is the ratio between the frequency of the oscillator (TimerOsc) and the desired clock frequency (NominalFreq). FreqDivRatio is a design constant chosen to be greater than 1.0001. The ADDEND value is added to the 32-bit accumulator register at every rising edge of the oscillator clock (TimerOsc). The clock counter is incremented at every carry pulse of the accumulator. Only one of this register is required for the entire group of eTSEC. Figure 31-13 shows the TMR_ADD register.

	0	15
Field	ADDEND	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr		

	16	31
Field	ADDEND	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x4814	

Figure 31-13. TMR_ADD Register

Table 31-15 describes the TMR_ADD fields.

Table 31-15. TMR_ADD Field Descriptions

Bits	Name	Description
0–31	ADDEND	Timer drift compensation addend register value. It is programmed with a value of 2^32/FreqDivRatio. For example: FreqOsc = 50 MHz FreqClock = 40 MHz FreqDivRatio = 1.25 RTCAD = ceil(2^32/1.25) = 0xCCCC_CCCD

31.9.5 Timer Accumulator register (TMR_ACC)

The Timer Accumulator register accumulates the value of the addend register into it. An overflow pulse of the accumulator is used to increment the timer clock by TMR_CTRL[TCLK_PERIOD]. This register is read only.

Figure 31-14 shows the TMR_ACC register.

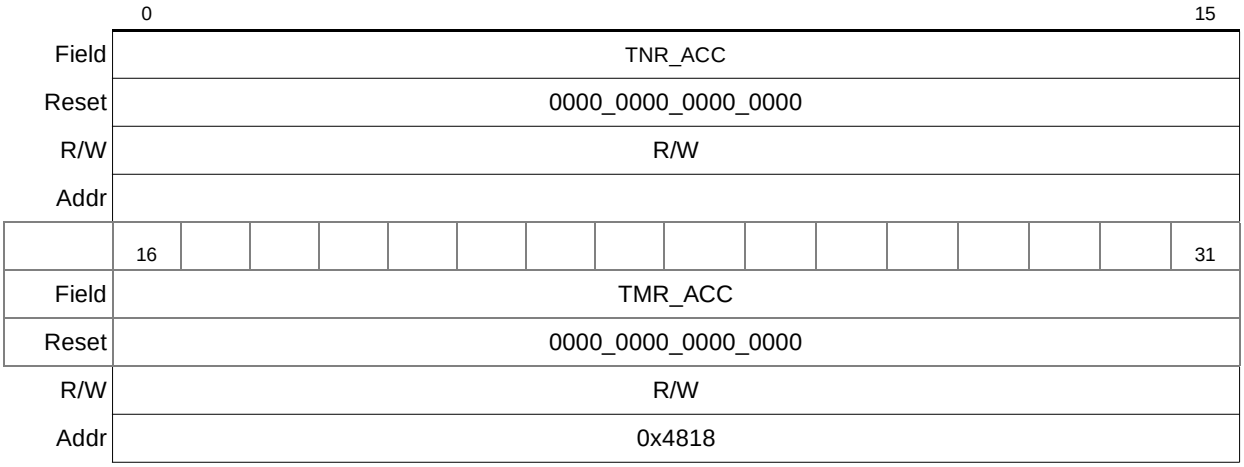


Figure 31-14. TMR_ACC Register

Table 31-16 describes the TMR_ACC fields.

Table 31-16. TMR_ACC Field Descriptions

Bits	Name	Description
0–31	TMR_ACC	32-bit timer accumulator register

31.9.6 Timer Prescale Register (TMR_PRSC)

The Timer prescale register is used to adjust output clock frequency. Figure 31-15 shows the TMR_PRSC register.

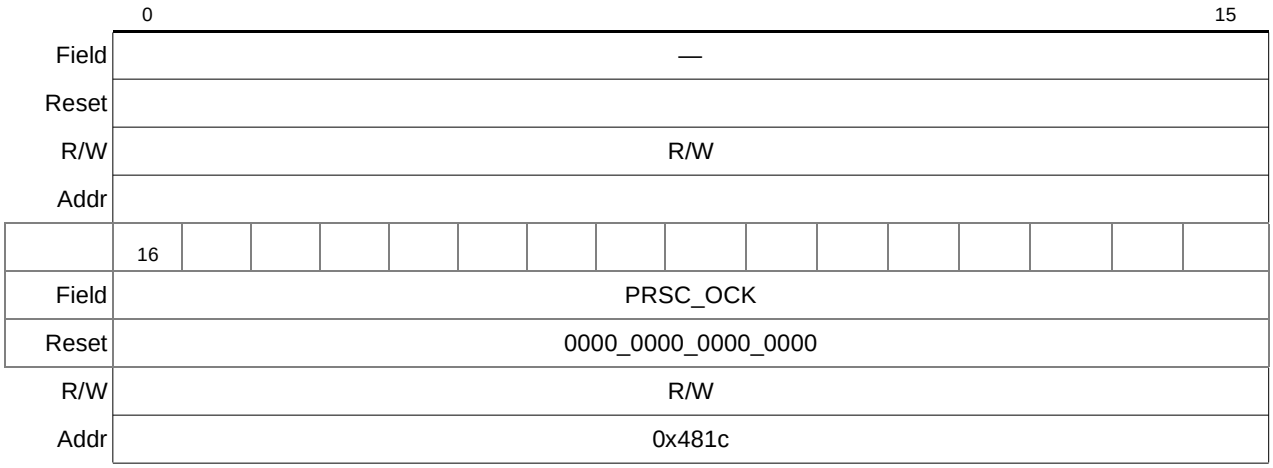


Figure 31-15. Prescale Register (TMR_PRSC)

Table 31-17 describes the TMR_PRSC fields.

Table 31-17. Prescale Register (TMR_PRSC)

Bits	Name	Description
0-15	—	RESERVED
16-31	PRSC_OCK	Output clock division/prescale factor. Output clock is generated by dividing the timer input clock by this number. Setting the prescaler value to “1” will generate an output clock in the same frequency as the system timer register.

31.9.7 Timer Offset Register (TMROFF_L/TMROFF_H)

The TMROFF register is shown in Figure 31-16.

The timer offset register is used to provide current time by adding its value to the clock counter.

	0															15
Field	TMROFF															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr																
	16													29	30	31
Field	TMROFF															
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x4820/0x4824															

Figure 31-16. TMROFF_L/TMROFF_H Register

Table 31-18 describes the fields.

Table 31-18. TMROFF_L/TMROFF_H Field Descriptions

Bits	Name	Description
0–31	TMROFF	The TMROFF, which is determined by the S/W calculations, updates the real time counter with its value. The update is executed when the H/W detects write transaction to the offset high register.

31.9.8 Alarm Time Register (TMR_ALARMn_L/TMR_ALARMn_H)

The TMR_ALARMn_L/TMR_ALARMn_H register is shown in Figure 31-17.

The Alarm Time comparator register (TMR_ALARMn_H/L) holds alarm time for comparison with the current time counter.

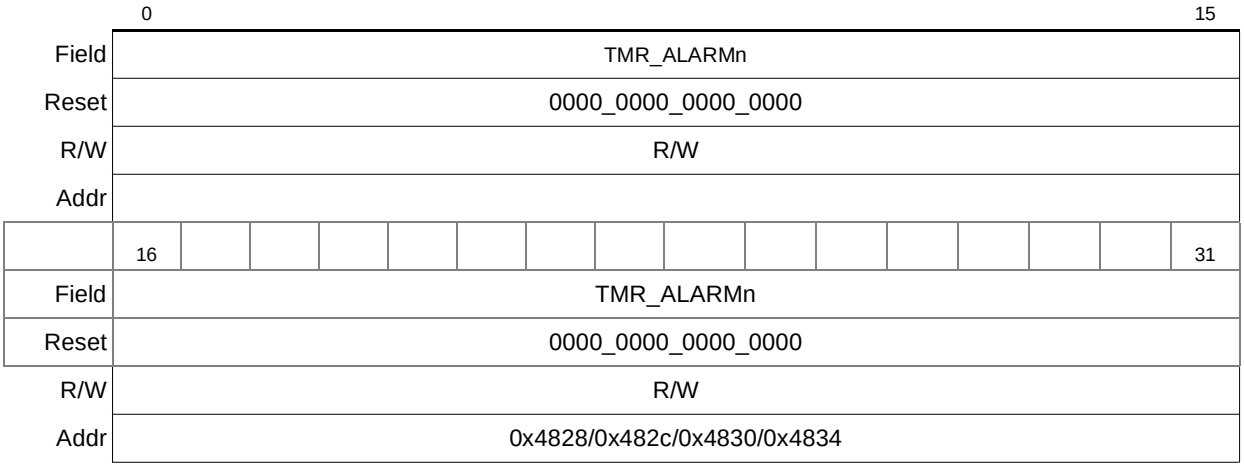


Figure 31-17. TMR_ALARMn Register Structure

Table 31-19 describes the TMR_ALARMn fields.

Table 31-19. TMR_ALARMn Bit Setting

Bits	Name	Description
0–31	TMR_ALARMn	The IEEE1588 RTC interrupt is generated when the value of the RTC counter (TMR_CNT_L/TMR_CNT_H) is greater than or equal to TMR_ALARMn[TMR_ALARM]. This value should be configured before enabling the timer. In FS mode, the alarm trigger is used as an indication to the fiper to start down counting. Only alarm 1 supports this mode. In FS mode, the alarm polarity bit should be configured to 0 (rising edge).

31.9.9 Timer Fixed Interval Period Register (TMR_FIPERn)

The Timer Fixed Interval Period pulse generator register is used to generate periodic pulses. This register is reset with 0xFFFF_FFFF to prevent any false pulse upon initialization. The down count register loads the value programmed in the fixed period interval (FIPER). The FIPER register must be programmed before the timer is enabled. In FS mode the FIPER should be programmed before the alarm value which triggers the start of the down counting. At every tick of the FreqOsc, the counter decrements by the value of TMR_CTRL[TCLK_PERIOD]. It generates a pulse when the down counter value reaches zero. It reloads the down counter in the cycle following a pulse. The user is responsible for loading these registers with desired period intervals. There are three FIPER registers.

Figure 31-18 shows the TMR_FIPER register.

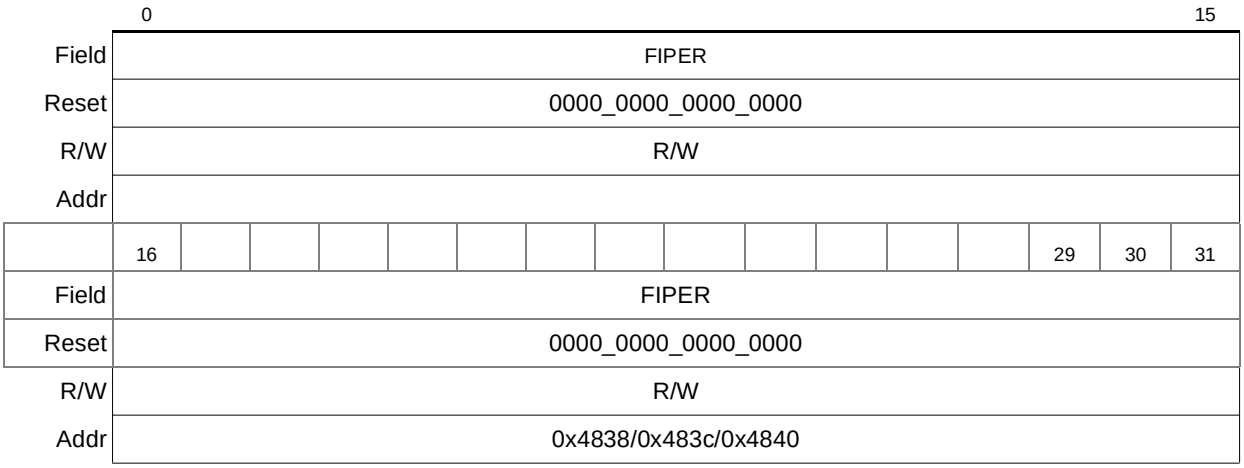


Figure 31-18. TMR_FIPERn Register

Table 31-20 describes the TMR_FIPERn fields.

Table 31-20. TMR_FIPERn Field Descriptions

Bits	Name	Description
0–31	FIPER	Fixed interval pulse period register. Note: In case the PPS signals should be phased aligned to the prescale output clock, the alarm value should be configured to **3 clock periods less** than the targeted value. In order to keep tracking the prescale output clock, each time before enabling the fiper, the user must reset the fiper by writing a new value to the register. The ratio between the prescale register value and the fiper value should be devisable by the clk period. **FIPER_VALUE = (prescale_value * tclk_per * N) - tclk_per** For example: prescale = 9 clock period = 10 fiper can get the following values: 80, 170, 260 .....

31.9.10 External Trigger Stamp Register (TMR_ETTSn_L/TMR_ETTSn_H)

The general purpose External Trigger Stamp register (TMR_ETTSn_H/L) holds the time at the programmable edge of the external trigger. There will be only two of these registers.

Figure 31-19 shows the TMR_ETTSn_H/L register.

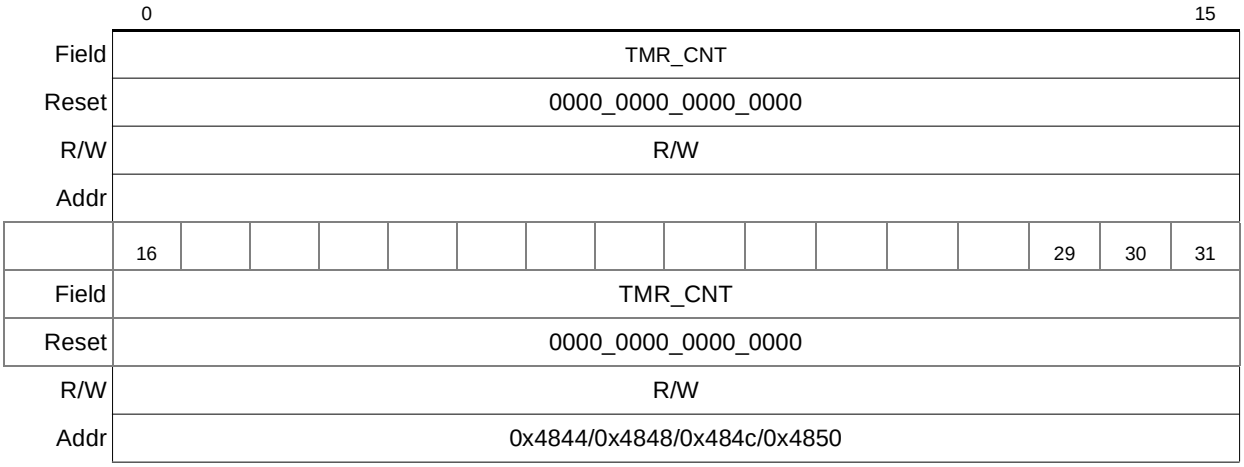


Figure 31-19. TMR_ETTSn_L/TMR_ETTSn_H Register

Table 31-21 describes the fields.

Table 31-21. TMR_ETTSn_L/TMR_ETTSn_H Field Descriptions

Bits	Name	Description
0–31	ETTS	Time stamp field at the programmable edge of the external trigger.

31.10 Time Stamp Unit

The TSU (Time Stamp Unit) logic implementation supports two different ways of operation. In-band and out-of-band methods. As mentioned in the preceding sections, the hardware assumption is that this is running on an Ethernet system.

In case of in-band mode, the timestamp reporting to the software is done in-band with the frame data. The user is responsible to set a dedicated PTP bit in Tx buffer descriptor when a PTP frame is transmitted. On the receiver side, the Ethernet mac sets the PTP bit in the Rx buffer descriptor following the TSU PTP recognition.

In case of out-of-band method, the reporting is done using dedicated registers per interface. The SW will read the transmitter’s timestamps after the “last” indication and the receiver’s timestamps if PTP bit was set in the RxBD. If there is sufficient time to guarantee that SW will be able to read time stamp register between PTP frames, then the out-of-band method is sufficient.

The TSU logic supports MII/RMII/GMII/RGMII interfaces.

In order to snapshot a precise time stamp, one of the hardware roles is to observe the MII ports, detect an SFD, and then capture the exact time.

Figure 31-20 shows the timestamp point of a PTP frame.

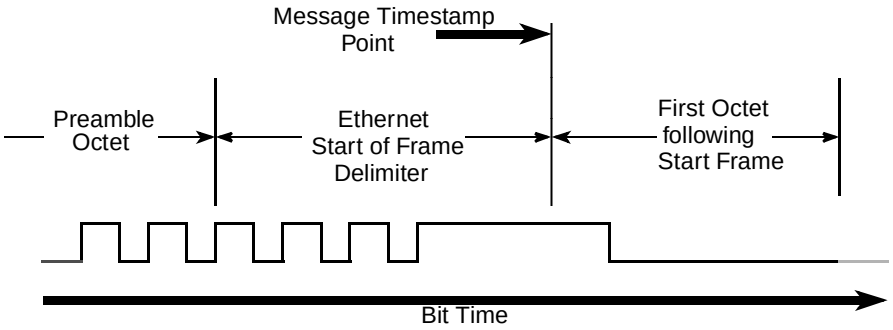


Figure 31-20. Message Timestamp Point

In addition, the TSU detects a PTP (Precise time protocol) frame of specific type and generates an indication to the CPU and to the MAC.

In the Version 1 specification, there are five types of PTP frame:

- Sync message
- Follow_Up message
- Delay_req message
- Delay_resp message
- Management message

Those five messages are divided into two groups:

- PTP event
- PTP general

The PTP event group consists of Sync and Delay_req messages. These messages should be timestamped and can cause an interrupt generation. The PTP general group consists of the rest of the frames, those frames transfer timestamps and determine general system parameters.

In order to detect a PTP frame, Table 31-22 summarizes the parsing fields and values:

Table 31-22. Parsing Values

Layer name	octet number	field name	value
IP	12-13	IP datagram	0x0800
UDP	23	Protocol: UDP	0x11
PTP event	36-37	Destination port number	0x13F
PTP general	36-37	Destination port number	0x140
Sync	74	control	0x0
Delay_req	74	control	0x1
Follow_up	74	control	0x2

Table 31-22. Parsing Values (continued)

Layer name	octet number	field name	value
Delay_resp	74	control	0x3
Management	74	control	0x4

In addition, the frames' lengths are determined in the following, Table 31-23.

Table 31-23. PTP frames length

message	frame length (octets)
Sync	171
Delay_req	171
Follow_up	97
Delay_resp	105
Management	105 + parameter_length/2

31.10.1 PTP Event Interrupts

In the case of out-of-band mode, the user can use the TSU (Time stamp unit) interrupts as a trigger to read the timestamps registers.The SW should clear the event register and read the timestamp before the next PTP frame detection. If a new PTP frame arrives and the relevant bit in the event register still set, an overrun indication will be asserted.

31.11 IEEE1588 Timer (Real Time Clock)

In a distributed control system containing multiple clocks, individual clocks tend to drift apart due to instabilities inherent in source oscillators and environmental conditions such as temperature, air circulation, mechanical stresses, vibration, aging etc. Hence, some kind of correction is necessary to synchronize individual clocks to maintain the notion of global time, which is accurate to some requisite clock resolution.

In order to achieve this goal, the IEEE1588 RTC supports the following:

- The nominal increment is chosen according to the nominal oscillator frequency.
- The drift is compensated by slightly increasing or decreasing the increment

Figure 31-21 illustrates the block diagram of the IEEE1588 RTC.

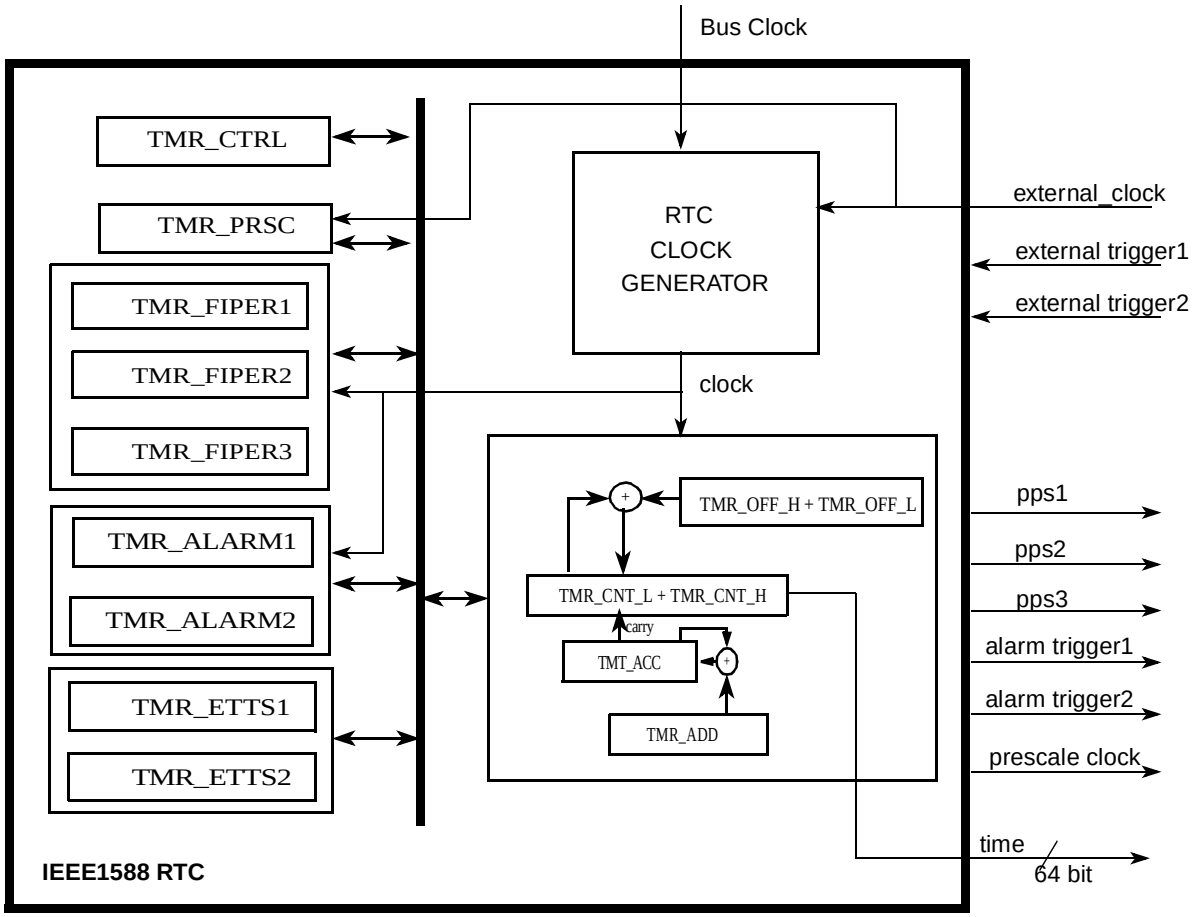


Figure 31-21. IEEE1588 RTC Block Diagram

All of the constituents of the frequency compensated clock operate at the frequency of input oscillator or sys_clk. The frequency compensation value contained in addend register is added to accumulator once every 1/sys_clk or 1/external_osc. The clock counter is incremented whenever the accumulator asserts an overflow pulse signal. Therefore the nominal frequency at which the clock counter is incremented is the frequency that the user had determined by updating the addend with a frequency compensation value.

The update of the frequency compensation value can be done every cycle of clock synchronization by sync and delay messages.

Table 31-24 shows examples of possible IEEE1588 RTC parameters. The results have been calculated according to the this equation:

frequency_compensation_value = 2^32/frequency_ratio.

Table 31-24. Values Example

Frequency sys_clk	Frequency Counter	Frequency Ratio	Frequency Compensation value
200MHz	150MHz	1.33	0xC07B3013
200MHz	190MHz	1.05	0xF3CF3CF3
200MHz	70MHz	2.86	0x5982AF70

In addition to the frequency compensation, the RTC enables updating of the offset of the timer counter. The offset value is derived from the timestamps transferred on PTP frames.

The S/W uses the offset register in order to add the correction value. The timer counter adds the value of the offset register if a write transaction to the relevant address is detected. The values in the offset register and in the addend register are represented in 2's complement.

The standard demands sub-micro resolution. The IEEE1588 RTC contains a 64-bit timer register and a 32-bit addend register. If the system clock reaches 200 MHz, by configuring the addend to 0xFFFF_FFFF, the counter will tick every 5 ns.

IEEE1588 RTC supports the following features:

- Two alarm output triggers, asserted if the timer value reaches the alarm register
- Three pulses per second output signals, generated by configuring the fiper registers
- Two input triggers in order to capture timestamps in a dedicated register
- Divided output clock, by configuring the prescale register

31.11.1 RTC Clock Sources

The IEEE1588 RTC enables three sources of input clock: the system clock, an external oscillator and Ethernet physical interface clock.

All the clocks should meet the performances and requirements of the standard.

The system clock could be up to 250MHz and might be sourced from the Baud Rate Generator in order to allow the user to generate a divided clock for the IEEE1588 timer.

31.11.2 Prescale Output Clock and Pulse per second Phase Alignment

In order to generate pulse per second signals that are phase aligned to the output prescale clock, as shown in Figure 31-22, the user should do the following:

1. Use the start fiper mode (TMR_CTRL[FS]) in the control register

2. The ratio between the prescale clock divider value and the fiper value should be devisable by the clock period, see Section 31.9.9, "Timer Fixed Interval Period Register (TMR_FIPERn)."

3. Configure the ALARM_1 register to a value which is less by 3 clock periods of the required FIPER enabling time.

4. Note that each time when writing a new value to the ALARM_1 when TMR_CTRL[FS] is set, the user should also write to the FIPERx register with the required PPS interval value.

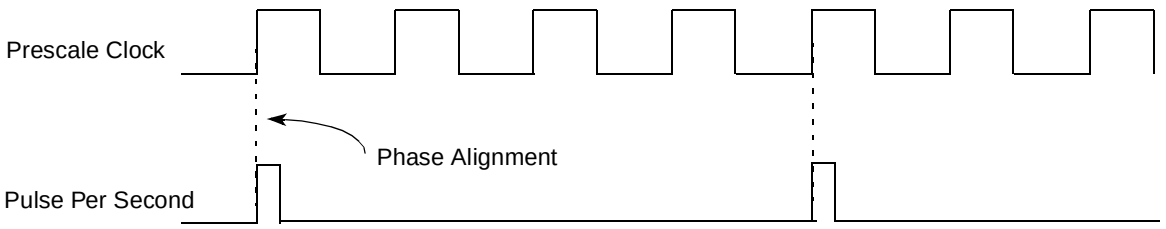


Figure 31-22. PPS and Prescale Clock Phase Alignment

31.11.3 External Signals Description

Table 31-25 describes the IEEE1588 Real Time Clock external signals:

Table 31-25. 1588 RTC External Signals

Signal Name	I/O	Description	Parallel Port Signal Pins	
			Primary Option	Secondary Option
PTP_PPS1	Output	Pulse per second output signal generated by configuring the FIPER1 register. Each time the FIPER1 value is expired one RTC clock period pulse is generated	CE_PC21	CE_PE20
PTP_PPS2	Output	Pulse per second output signal generated by configuring the FIPER2 register. Each time the FIPER2 value is expired one RTC clock period pulse is generated	CE_PC17	CE_PD19
PTP_PPS3	Output	Pulse per second output signal generated by configuring the FIPER3 register. Each time the FIPER3 value is expired one RTC clock period pulse is generated	CE_PC5	CE_PF29
PTP_ALARM1	Output	Alarm output triggers, set if the timer value reaches the alarm1 register	CE_PC0	CE_PE14
PTP_ALARM2	Output	Alarm output triggers, set if the timer value reaches the alarm2 register	CE_PC23	CE_PE6
PTP_REF_CLK	Output	Divided output clock, by configuring the TMR_PRSC register	CE_PC12	CE_PE26
PTP_CLK	Input	External oscillator Real Time Clock	CE_PC30	
PTP_EXT_TRIG1	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS1L/TMR_ETSS1H	CE_PB4	CE_PC14
PTP_EXT_TRIG2	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS2L/TMR_ETSS2H	CE_PC11	CE_PC27

31.12 PTP Frame Reception

The following describes the main actions performed for each operation mode when a PTP frame was received.

31.12.1 In-Band Mode

To use in-band mode, configure the UCC as follows:

- UCC in Ethernet Mode (GUMR[MODE])
- Set IEEE1588 mode in UPSMR[PTPE]
- Set MACFG[SRP] - Soft Rx Preamble mode

In order to work in in-band mode, the PTP should set the in-band mode in the TSMR register.

In this mode the received timestamps are transferred as part of the data payload.

The receive TSU (time stamp unit) is monitoring the received Ethernet frame on the physical line. When SFD is detected, the IEEE1588 Rx TSU locks the Real Time Value of the IEEE1588 clock and then notifies the Ethernet MAC that SFD is detected and the timestamp associated with it is ready.

The Ethernet MAC stores the 64-bit timestamp in memory as part of the frame, at the first eight bytes of each Ethernet frame, regardless of PTP indication.

When the Rx TSU parse unit is detecting a PTP frame, it will notify the Ethernet MAC that a PTP frame has been detected and the Ethernet controller will set RxBD[RPTP]

When the Software detects that a PTP frame has arrived, by getting an interrupt or by polling the RxBD[RPTP], it will search for the PTP frame type and read the timestamp located in the first eight bytes of each frame in memory if needed.

In case the PTP frame is marked with CRC, Overrun, or Non-octet receive errors, the interrupt in the TMR_PEVENT will not be set and the software should drop this frame regardless of the indication in the RxBD[RPTP].

31.12.2 Out-of-Band Mode

In order to work in out-of-band mode the UCC should be configured as followed:

- UCC in Ethernet Mode (GUMR[MODE])
- Set IEEE1588 mode in UPSMR[PTPE]

The receive TSU (time stamp unit) is monitoring the received Ethernet frame on the physical line. When SFD is detected, the IEEE1588 Rx TSU locks the Real Time Value of the IEEE1588 clock.

If the Rx TSU parse unit has detected a PTP frame, it will notify the Ethernet MAC that a PTP frame has been detected, and the Ethernet controller will set RxBD[RPTP]. In addition, the Rx TSU will set the corresponding bit in the event register and generate an interrupt to the CPU.

When the Software detects that a PTP frame has arrived by getting an interrupt or by polling the RxBD[RPTP] it will read the dedicated timestamp register (TMR_UCx_RXTS_H/L) through the host interface.

In case the PTP frame is marked with CRC, Overrun, or Non-octet receive errors, the interrupt in the TMR_PEVENT will not be set and the software should drop this frame regardless of the indication in the RxBD[RPTP].

31.13 PTP Frame Transmission

The following describes the main actions performed for each operation mode when a PTP frame is transmitted.

31.13.1 In-Band Mode

If the Software transmits a PTP frame, it should set the appropriate bit in the buffer descriptor TxBD[TPTP]. The Ethernet MAC is responsible to indicate to the Tx TSU that a PTP frame is transmitted.

The transmit TSU (time stamp unit) is monitoring the transmitted Ethernet frame on the physical line. When SFD is detected, the IEEE1588 Tx TSU locks the Real Time Value of the IEEE1588 clock. If the Tx TSU gets a PTP frame indication, it sets the corresponding bit in the event register and generates interrupt to the CPU.

The SW should access the dedicated timestamp register (TMR_UCx_TXTS_H/L) through the host interface after an interrupt generation or when the “last” bit in the TxBD is set (TxBD[last]).

31.13.2 Out-of-Band Mode

Same as in-band operation.

31.14 Initialization Sequence

TSU mode registers:

Set the PTP_TSMR[opmode] to in-band or out-of-band mode

Program the TMR_PEMASK register

Write 0xFFFF_FFFF to TMR_PEVENT to clear the PTP Event register

UCC mode register:

Set the GUMR[MODE] to Ethernet Mode

Select Ethernet PHY interface

- Select nibble or octal interface in MCCFG2[IFM] and UPSMR
- Set the UPSMR[PTPE] to one (enables IEEE1588 support)

Note: In RMII mode PTP_TSMR[opmode] must be configured first

Set MACCFG2[SRP] only when working in in-band mode

RTC mode registers:

Initialize the TMR_ALARMn registers to 0xFFFF_FFFF

Program the IEEE1588 timer clock period in TMR_CTRL[tclk_period]

Select the timer clock source in the TMR_CTRL[clksel]

Program the TMR_TEMASK register

Write 0xFFFF_FFFF to TMR_TEVENT to clear the RTC Event register

Program the TMR_ADD according to the needed drift compensation needed value

Enable Sequence:

Enable the RTC by setting TMR_CTRL[TE]

Enable the PTP by setting PTP_TSMR[En]

Enable the UCC by setting GUMR[ENR]

Enable the UCC by setting GUMR[ENT]

Chapter 32
ATM Controller AAL0, AAL1, and AAL5

32.1 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, AAL1 circuit emulation service (CES), AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.

For each virtual channel (VC), the controller's ATM pace control (APC) unit generates a cell transmission rate to implement constant bit rate (CBR), variable bit rate (VBR), 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 supports a special mode for ATM/TDM Interworking(IW). The QUICC Engine block performs automatic data forwarding between ATM channels and the MCC's TDM channels without core intervention.

The QUICC Engine block ATM SAR controller applications are as follows:

- ATM line card controllers
- ATM-to-WAN Interworking (frame relay, T1/E1 circuit emulation)
- Residential broadband network interface units (NIU) (ATM-to-Ethernet)
- High-performance ATM network interface cards (NIC)
- Bridges and routers with ATM interface

32.1.1 Features

The ATM controller has the following features:

- Full duplex segmentation and reassembly at OC12 rate.
- UTOPIA level II master and slave modes 8/16 bit
- AAL5, AAL1, AAL2, 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
- Special mode for ATM-to-TDM or ATM-to-ATM data forwarding
- CLP and congestion indication marking
- User-defined cells up to 65 bytes
- Separate TxBD and RxBD tables for each virtual channel (VC)
- Special mode of global free buffer pools for dynamic and efficient memory allocation with early packet discard (EPD) support
- Interrupt report per channel using four priority interrupt queues
- Compliant with ATMF UNI 4.1 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
- AAL1 cell format
  - — Reassembly
    - – Reassemble PDU directly to external memory
    - – Support for partially filled cells (configured on a per-VC basis)
    - – Sequence number check
    - – Sequence number protection (CRC-3 and parity) check
  - — Segmentation
    - – Segment PDU directly from external memory
    - – Partially filled cells support (configured on a per-VC basis)
    - – Sequence number generation
    - – Sequence number protection (CRC-3 and even parity) generation
  - — Structured AAL1 cell format
    - – Automatic synchronization using the structured pointer during reassembly

  - Structured pointer generation during segmentation
  - Unstructured AAL1 cell format
    - Clock recovery using external SRTS (synchronous residual time stamp) logic during reassembly
    - SRTS generation using external logic during segmentation
- AAL0 format
  - Receive
    - Whole cell is put in memory
    - Only cell payload is saved in memory
    - CRC10 pass/fail indication
  - Transmit
    - Reads a whole cell from memory
    - CRC10 insertion option
- AAL1 circuit emulation service
  - Refer to Chapter 35, “ATM AAL1 Circuit Emulation Service.
- AAL2 format
  - Refer to Chapter 41, “E2AAL2 Microcode”
- 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 8/16 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

32.1.2 ATM Controller—Modes of Operation

The ATM Controller main modes of operations are:

- ATM SAR AAL5, AAL1, AAL2, 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
- ATM-to-TDM Interworking
- CAS Support
- UPC
- Operation in a 82xx compatible mode and in Multi-Threading mode for high performance/ bandwidth requirements.

32.2 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.

32.2.1 Transmitter Overview

Before the transmitter is enabled, the host must initialize the QUICC Engine block and create the transmit data structure, described in Section 32.3, “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*[0–15] when using UTOPIA 16 and 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.

Unlike the 82XX family the device doesn't transmit Idle cells and operation is in what is called in the 82XX user's manual "internal rate mode"

32.2.1.1 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 32.3.4.2, "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.

32.2.1.2 AAL1 Transmitter Overview

The QUICC Engine block supports both structured and unstructured AAL1 formats. For the unstructured format, the transmitter reads 47 bytes from the external buffer and inserts them into the AAL1 user data field. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is generated and inserted into the cell. The QUICC Engine block

supports synchronous residual time stamp (SRTS) generation using external PLL. If this mode is enabled, the QUICC Engine block reads the SRTS code from the external logic and inserts it into four outgoing cells.

For the structured format, the transmitter reads 47 or 46 bytes from the external buffer and inserts them into the AAL1 user data field. The QUICC Engine block generates the AAL1 PDU header and inserts it into the cell. The header consists of the SN, SNP, and the structured pointer.

The QUICC Engine block supports partially filled cells configured on a per-VC basis. In this mode (TCT[PFM] = 1), only valid octets are copied from the buffer to the ATM cell; the rest of the cell is filled with padding octets.

32.2.1.2.1 AAL1 CES Transmitter Overview

Refer to CES chapter, Section 35.2, “AAL1 CES Transmitter Overview.”

32.2.1.3 AAL0 Transmitter Overview

No specific adaptation layer is provided for AAL0. The ATM controller reads a whole cell from an external buffer, which always contains exactly one AAL0 cell. The ATM controller optionally generates CRC10 on the cell payload and places it at the end of the payload (CRC10 field). AAL0 mode can be used to send OAM cells or AAL3/4 raw cells.

32.2.1.4 AAL2 Transmitter Overview

Refer to AAL2 chapter, Section 32.2.1, “Transmitter Overview.”

32.2.2 Receiver Overview

Before the receiver is enabled, the host must initialize the QUICC Engine block and create the receive data structures described in Section 32.3, “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 32.3.9, “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).

32.2.2.1 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.

32.2.2.2 AAL1 Receiver Overview

The ATM controller supports both AAL1 structured and unstructured formats. For the unstructured format, 47 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is checked. The QUICC Engine block supports SRTS clock recovery using an external PLL. In this mode, the QUICC Engine block tracks the SRTS from the four incoming cells and writes the SRTS code to external logic. See Section 32.2.22.4.1, “SRTS Generation and Clock Recovery Using External Logic.”

In the unstructured format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and starts receiving. If an SN mismatch is detected, the receiver closes the RxBD, sets the sequence number error flag (RxBD[SNE]), and switches to hunt state, where it stays until a cell with a valid SN field is received.

For the structured format, 47 or 46 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of SN and SNP, is checked and the PDU status is written to the BD.

In the structured format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer to gain synchronization. When one arrives, the receiver leaves the hunt state and starts receiving. Then the receiver opens a new buffer. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer. If an SN mismatch is detected, the ATM receiver closes the current RxBD, sets RxBD[SNE], and returns to the hunt state. The receiver then waits for a cell with a valid structured pointer to regain synchronization.

The QUICC Engine block supports partially filled cells configured on a per-VC basis. In this mode (RCT[PFM] = 1), the ATM controller copies only the valid octets from the cell user data field to the buffer.

32.2.2.2.1 AAL1 CES Receiver Overview

Refer to Section 35.3, “AAL1 CES Receiver Overview.”

32.2.2.3 AAL0 Receiver Overview

For AAL0, no specific adaptation layer processing is done. The ATM controller copies the whole cell to an external buffer. Each buffer contains exactly one AAL0 cell. The ATM controller calculates and checks

the CRC10 of the cell payload and sets RxBD[CRE] if a CRC error occurs. AAL0 mode can be useful for receiving OAM cells or AAL3/4 raw cells. 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.

32.2.2.4 AAL2 Receiver Overview

Refer to Section 41.4, "AAL2 Receiver."

32.2.3 ATM Multi-Threading

The Multi-threading mode of operation is introduced in the QUICC Engine block in order to support higher ATM bit rates, up to OC-12 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 32-25).

Figure 32-1 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 32.3.1, "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 32.3.3, “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 32-20 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.

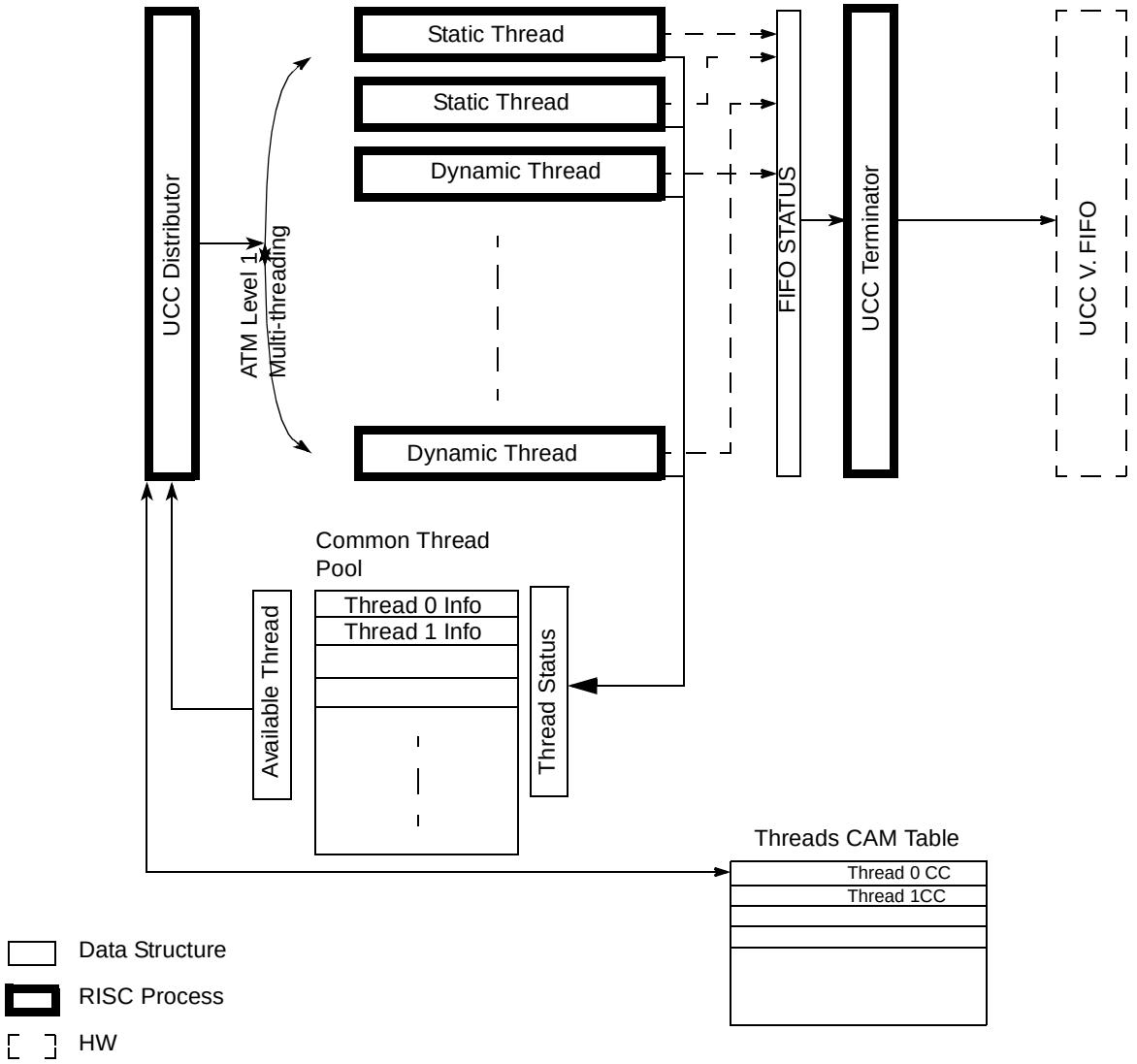


Figure 32-1. ATM UCC Multi-Threading Concept

In MSP HTCT mode it is not enough to use the HTCT-Channel Code as the only input for the multi thread distributor decision process, since in systems where there is only one HTCT the multi thread will not be useful. Therefore in addition to the HTCT CC there is also the FIFO ID index in the Distributor Tx CAM structure. In this case each CAM entry is of 4 bytes length. See Figure 32-2. Still in a system where there is only 1 HTCT and under it only one FIFO, then the multi thread is not useful for the Tx side.Potential problem- in the HTCT we keep information needed from next thread steps HTCT[FFS_TMP] and HTCT[EFDT_CUR] - this temporary parameters now should be stored in the thread page since the same HTCT could be used in several threads.

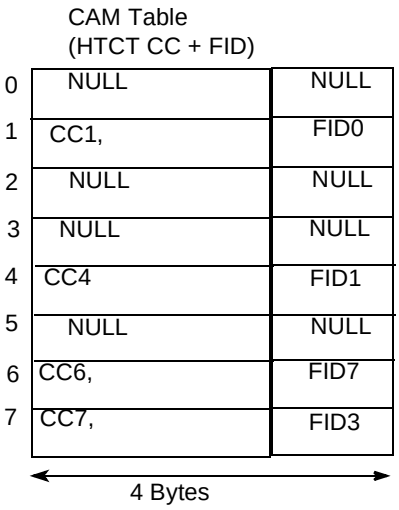


Figure 32-2. MSP HTCT distributor structures

In the same way for MSP Tx FIFO mode the multi thread algorithm can use the UCC no., PHY number and the FDT ID in order to perform the distributor process. In this case each CAM entry is still of 2 bytes length. See Figure 32-3.

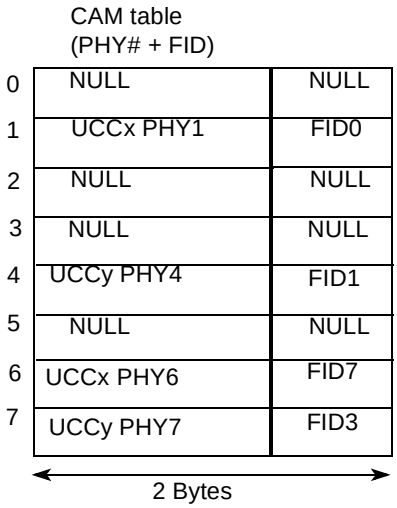


Figure 32-3. MSP FIFO Distributor Structures

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.

32.2.4 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 32.2.19, "Performance Monitoring."

32.2.5 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.

32.2.5.1 APC Modes and ATM Service Types

The ATM Forum (http://www.atmforum.com) defines the service types described in Table 32-1.

Table 32-1. 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 32.2.6, "ATM Traffic Type." For information about prioritization, see Section 32.2.11, "Determining the Priority of an ATM Channel."

32.2.5.2 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 32.3.6, "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 32-4. 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.

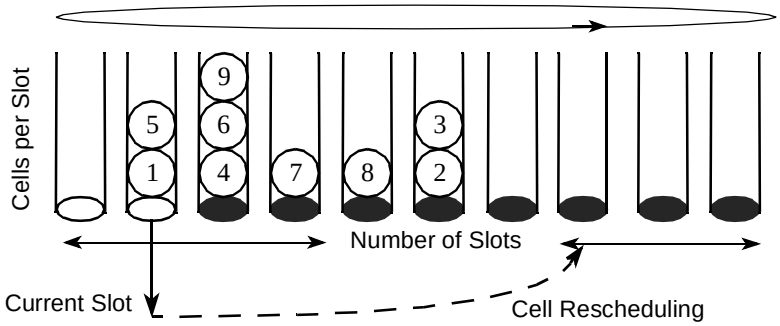


Figure 32-4. APC Scheduling Table Mechanism

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 32-5 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 it's parameters stored in a data structure named TCT- Transmit Connection Table and TCTE- Transmit Connection Table Extension.

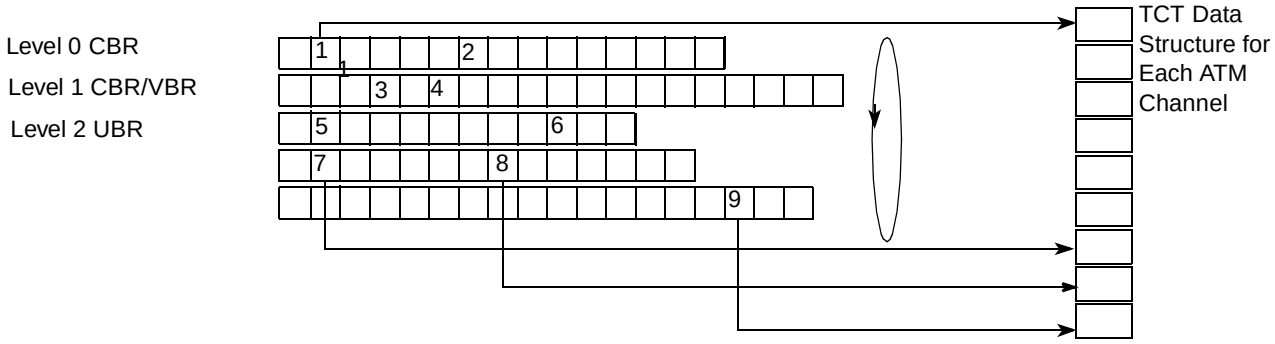


Figure 32-5. ATM Scheduling

32.2.5.3 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.

32.2.5.3.1 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.

32.2.5.3.2 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.

32.2.5.3.3 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.

$$\text{(C)} \qquad \text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}}$$

32.2.6 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 32.3.4.2, "Transmit Connection Table (TCT)."

32.2.6.1 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.

32.2.6.2 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

32.2.6.3 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 32-6. (The GCRA is also known as the leaky bucket algorithm.)

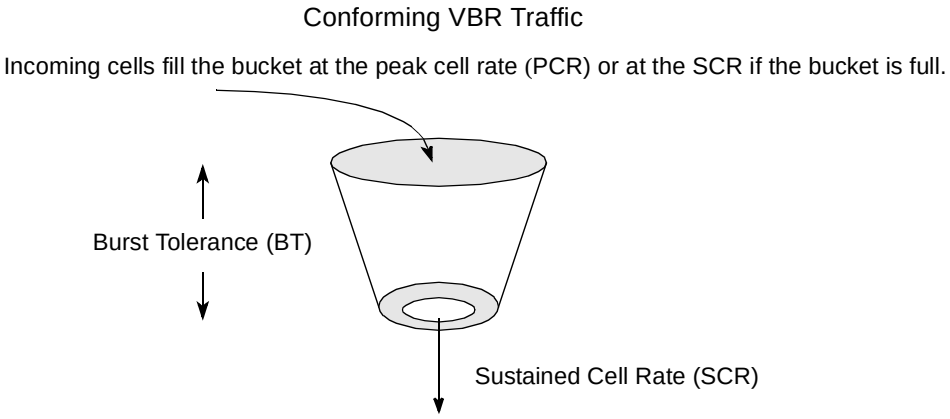


Figure 32-6. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

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.

32.2.6.3.1 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 32.2.5.3.3, "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.

$$\text{PCR [slots]} = (155.52 \text{ Mbps})/(6 \text{ Mbps} \times 8) = 3.24$$

$$3.24 = 3 + 0.24 \times 256/256 = 3 + 61.44/256 \sim 3 + 62/256$$

$$\text{PCR} = 3 \qquad \text{PCR\_FRACTION} = 62$$

$$\text{SCR [slots]} = (155.52 \text{ Mbps})/(2 \text{ Mbps} \times 8) = 9.72$$

$$9.72 = 9 + (0.72 \times 256/256) = 9 + 184.32/256 \sim 9 + 185/256$$

$$\text{SCR} = 9 \qquad \text{SCR\_FRACTION} = 185$$

Equation D yields the number of slots the user writes to the channel's TCT[BT].

(D) $$BT\ [slots] = (MBS[cells] - 2) \times (SCR[slots] - PCR[slots]) + SCR[slots]$$
$$= (1000 - 2) \times ((9+185/256) - (3+62/256)) + (9 +185/256)$$
$$= 6477$$

32.2.6.3.2 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 32.3.4.2.6, “VBR Protocol-Specific TCTE.”

32.2.6.4 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 32.3.4.2.7, “UBR+ Protocol-Specific TCTE***.”

32.2.7 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 32-47. 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.

32.2.7.1 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 i.e. 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 i.e. NOS-1 slots away from it's 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 it's 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.

32.2.8 APC Flux Compensation

For variable bit rate PHY (e.g. 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 32-45) a QUICC Engine Time-Stamp register, CETSR, is set up by programming the CETSCR register, see Section 20.3.9, "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 32-44. 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 32-46, 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 32-45. When the PHY is slowed down for a long period of time and has gained a large credit i.e. 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.

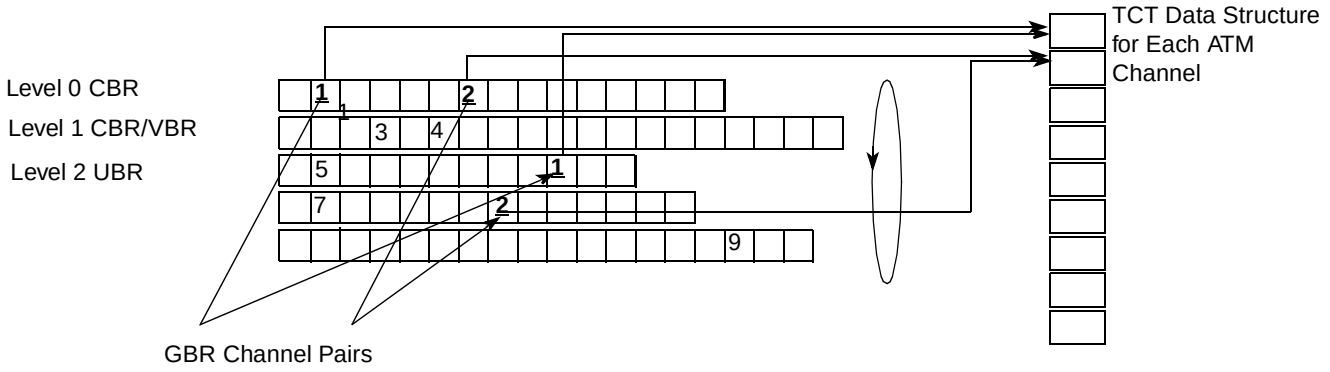
32.2.9 GBR Guaranteed Bit Rate

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.



32.2.10 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 32-7, for an example APC with 6 priority levels, of which 4 are UBR+.

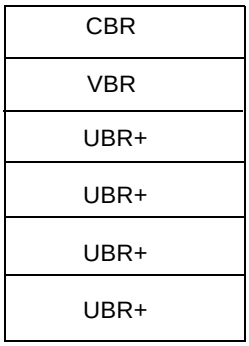


Figure 32-7. UBR+ Priority Level Example

The user must initialize the UBR+ priority decision table (see Figure 32-54) before enabling the UBR+ prioritized mode. This table is located at address Scheduler_Parameter_Table [AFP_LAST]+8. See Figure 32-8, 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.

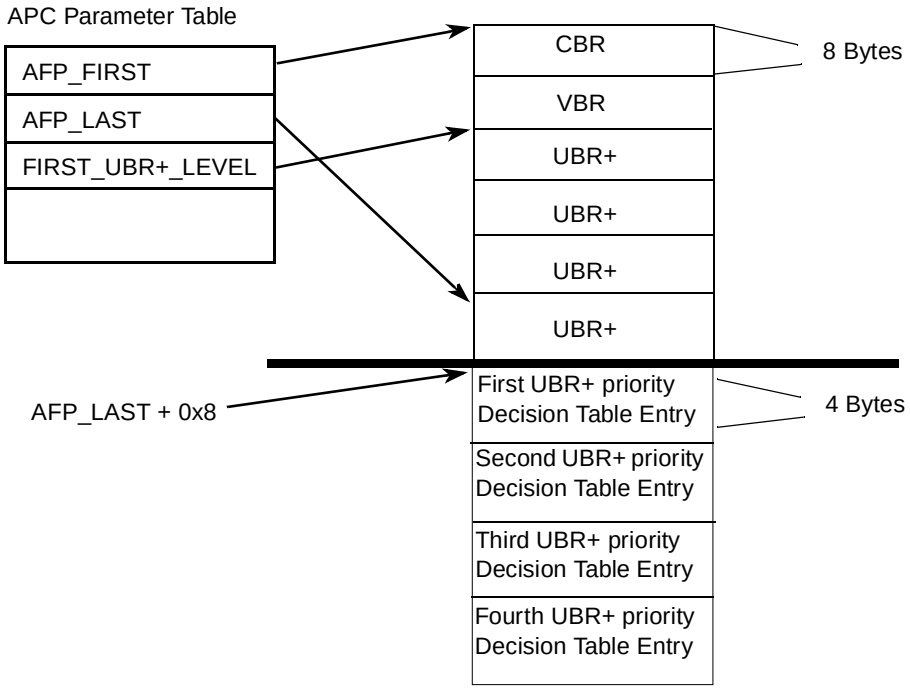


Figure 32-8. UBR+ Priority Table Example

32.2.11 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 32.3.6, "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 32.4.1, "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.

32.2.12 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.

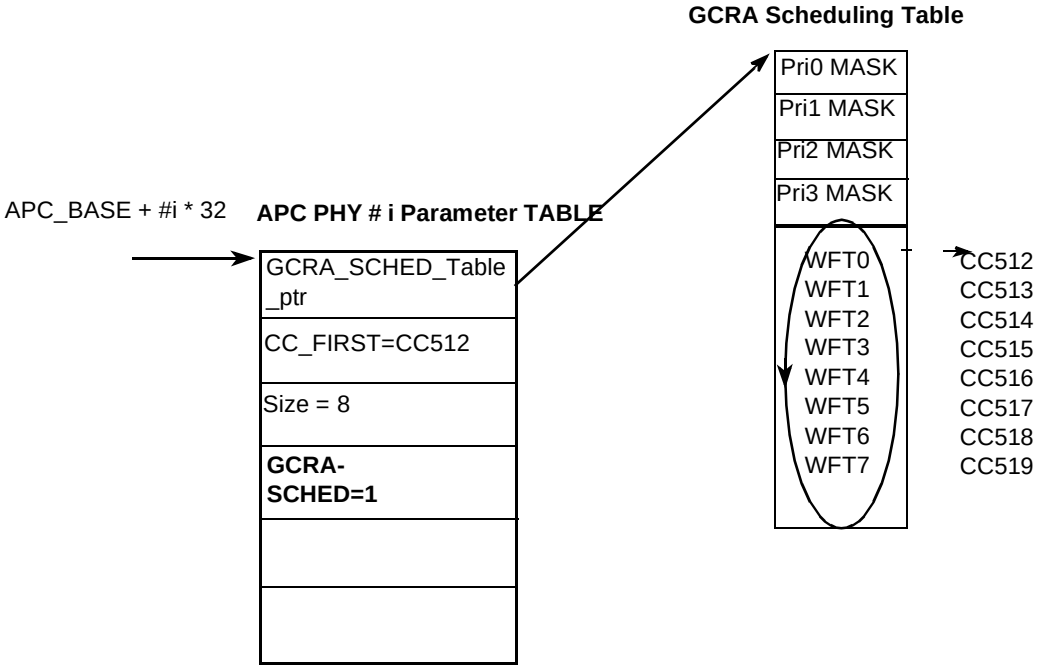


Figure 32-9. GCRA Scheduling structure

Figure 32-9 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.

The GCRA scheduler enables full utilization of the bandwidth in addition to the non work conserving GCRA. If no channel is eligible for transmission at a given time or in the case where the scheduler is empty there is another transmit queue which contains a list of channels for transmission. those are transmitted in a work conserving scheduling using round-robin algorithm for scheduling, This mode is useful for UBR channels. Transmission from this link is done without any comparison to the system time and without performing rescheduling to these channels.

For more information about the GCRA scheduler structures See Section 32.3.7, “GCRA Scheduler Structures”.

32.2.12.1 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 32-1. 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:

$INC\ (CH5)=424/(1 \times 10^{-6} \times 12 \times 10^{6})=35.3 => PCR=0x23,\ PCR\ FRAC=0x55$

$INC\ (CH6)=424/(1 \times 10^{-6} \times 32 \times 10^{3})==>PCR=0x33B5\ PCR\ FRAC=0x80$

32.2.12.2 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 32.4.1, “ATM Commands.” This command actually sets the corresponding bit in the Pri_MASK field.

32.2.13 Hierarchical Scheduling

32.2.13.1 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 32.3.8, “Hierarchical Scheduling.”

32.2.13.2 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 32.3.8, “Hierarchical Scheduling

32.2.14 VCI/VPI Address Lookup Mechanism

The QUICC Engine block supports three ways for address look up of incoming cells:

- Address compression see Section 32.2.14.1, “Address Compression”
- Mini-CAM address look-up see Section 32.2.14.2, “Mini-CAM Address Look-Up”
- Channel Code extracted from User Defined Cell header. see Section 32.2.16.1, “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.

32.2.14.1 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 32-10 shows an example of address compression.

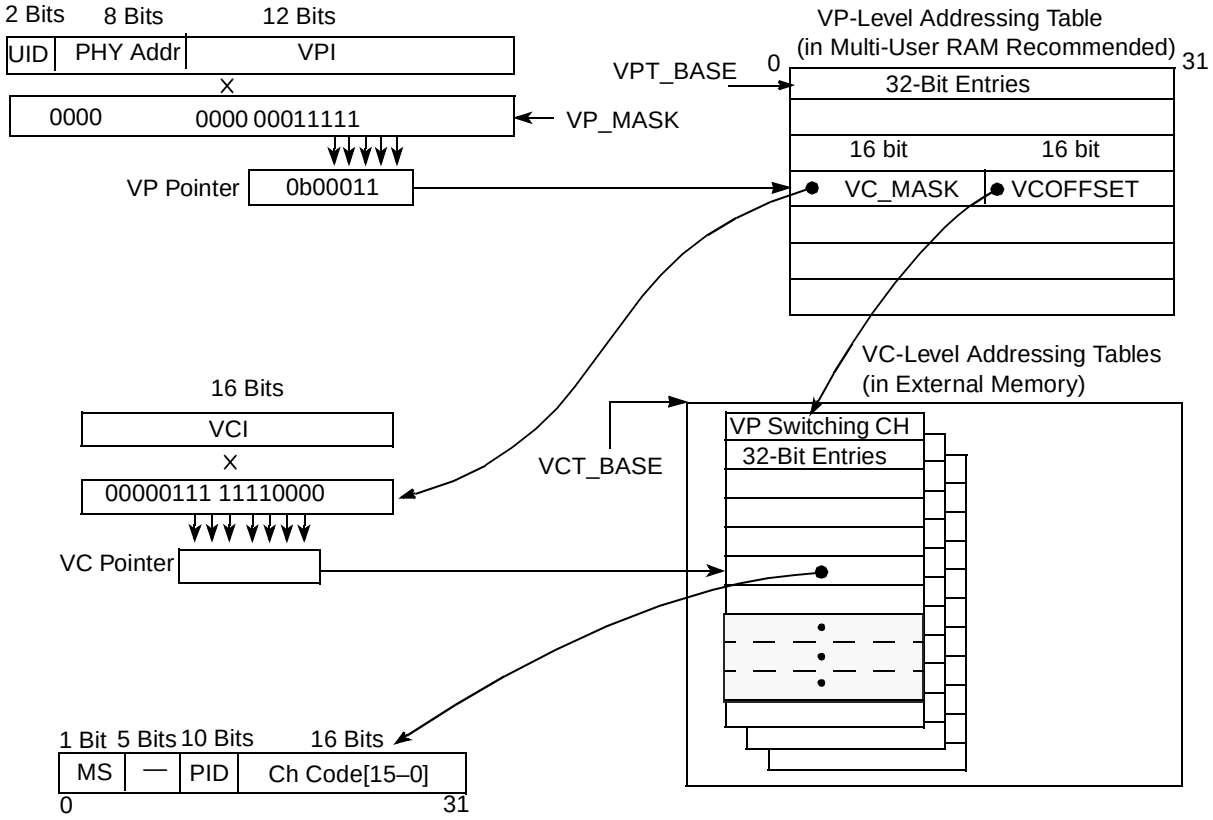


Figure 32-10. Address Compression Mechanism

Figure 32-10 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 32-2.

Table 32-2. 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.
0	MS	Match status. 0 Match was found. 1 Match was not found.

Table 32-2. Field Descriptions for Address Compression (continued)

Bits	Name	Description
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.

32.2.14.1.1 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 32-11 gives the general formula for determining VCOFFSET.

General formula: $VCOFFSET_{(n+1)} = VCOFFSET_n + 2^{(\text{number of ones in VC\_MASKn})}$

Figure 32-11. General VCOFFSET Formula for Contiguous VCLTs

Table 32-3 shows example VCOFFSET calculations for a VP-level table with four entries.

Table 32-3. 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 32-4 gives an example of VP-level table entry address calculation.

Table 32-4. 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 32-12 shows the VP pointer address compression from Table 32-4.

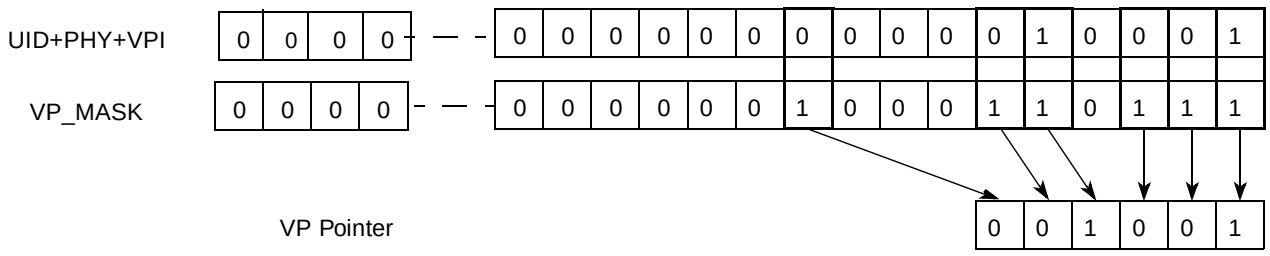


Figure 32-12. VP Pointer Address Compression

32.2.14.1.2 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 32-5. Note that VCOFFSET is assumed to be 0x100 for this example.

Table 32-5. 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 32-13 shows the VC pointer address compression from Table 32-5.

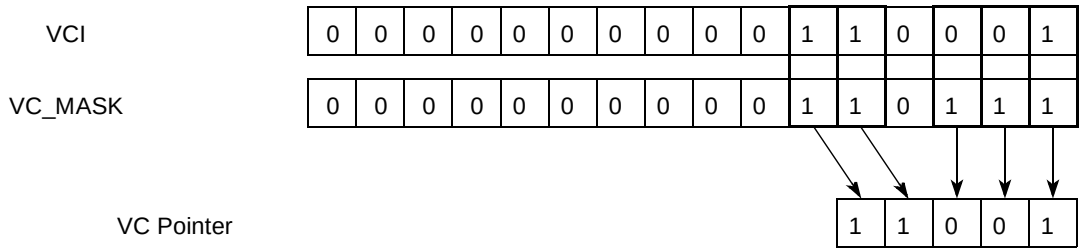


Figure 32-13. VC Pointer Address Compression

32.2.14.1.3 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 32.3.2.8, “Dynamic Address Compression Table change”.

32.2.14.2 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 32-14 illustrates the operation of this look-up

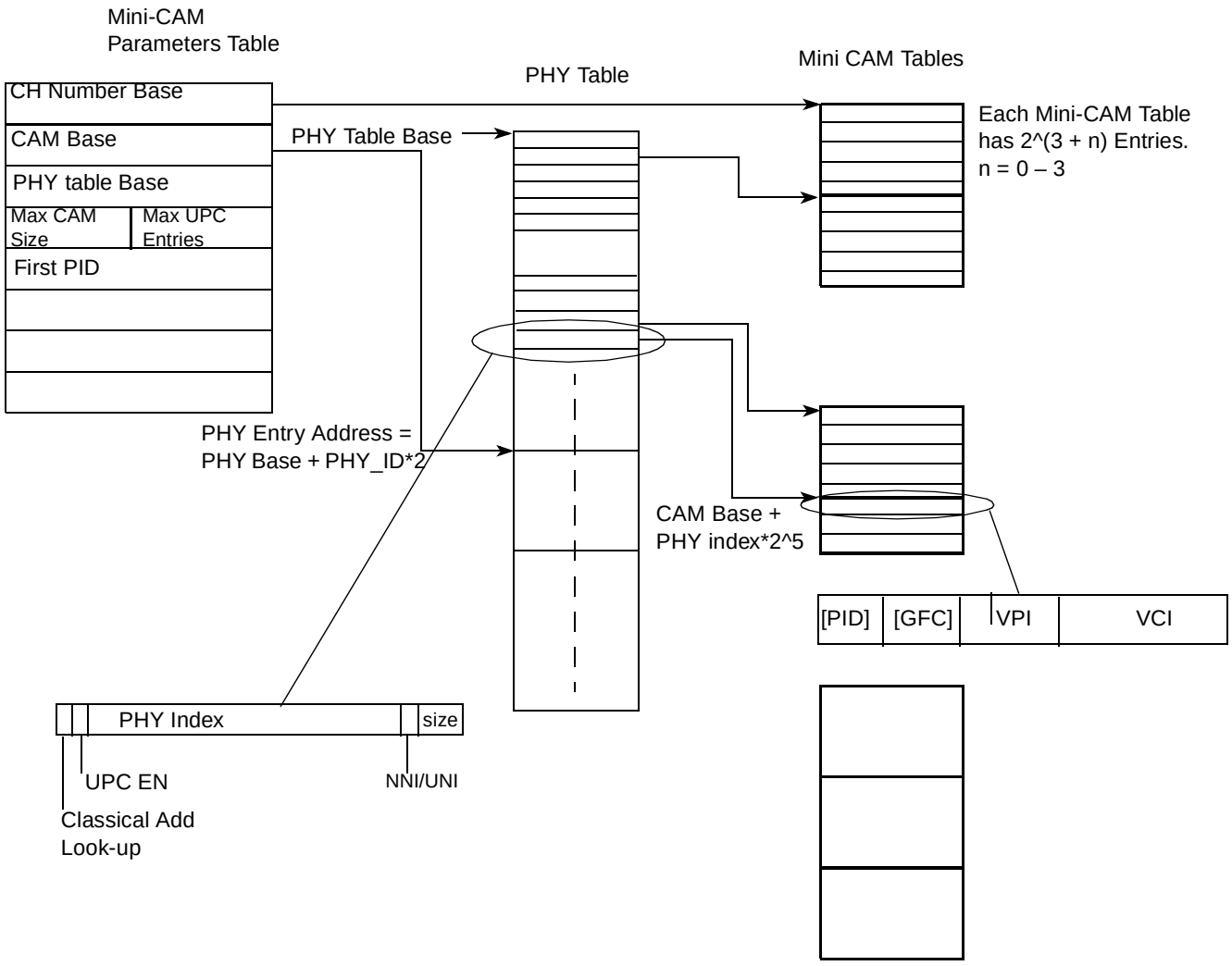


Figure 32-14. Mini-CAM Address Look-Up

The data structures used in this address look-up mode are described in the following paragraphs:

32.2.14.2.1 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 32-27. 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.

32.2.14.2.2 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 (i.e. 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.

32.2.14.2.3 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]

32.2.14.2.4 Mini-CAM Lookup Process

Figure 32-15 is a schematic flow of the look-up process.

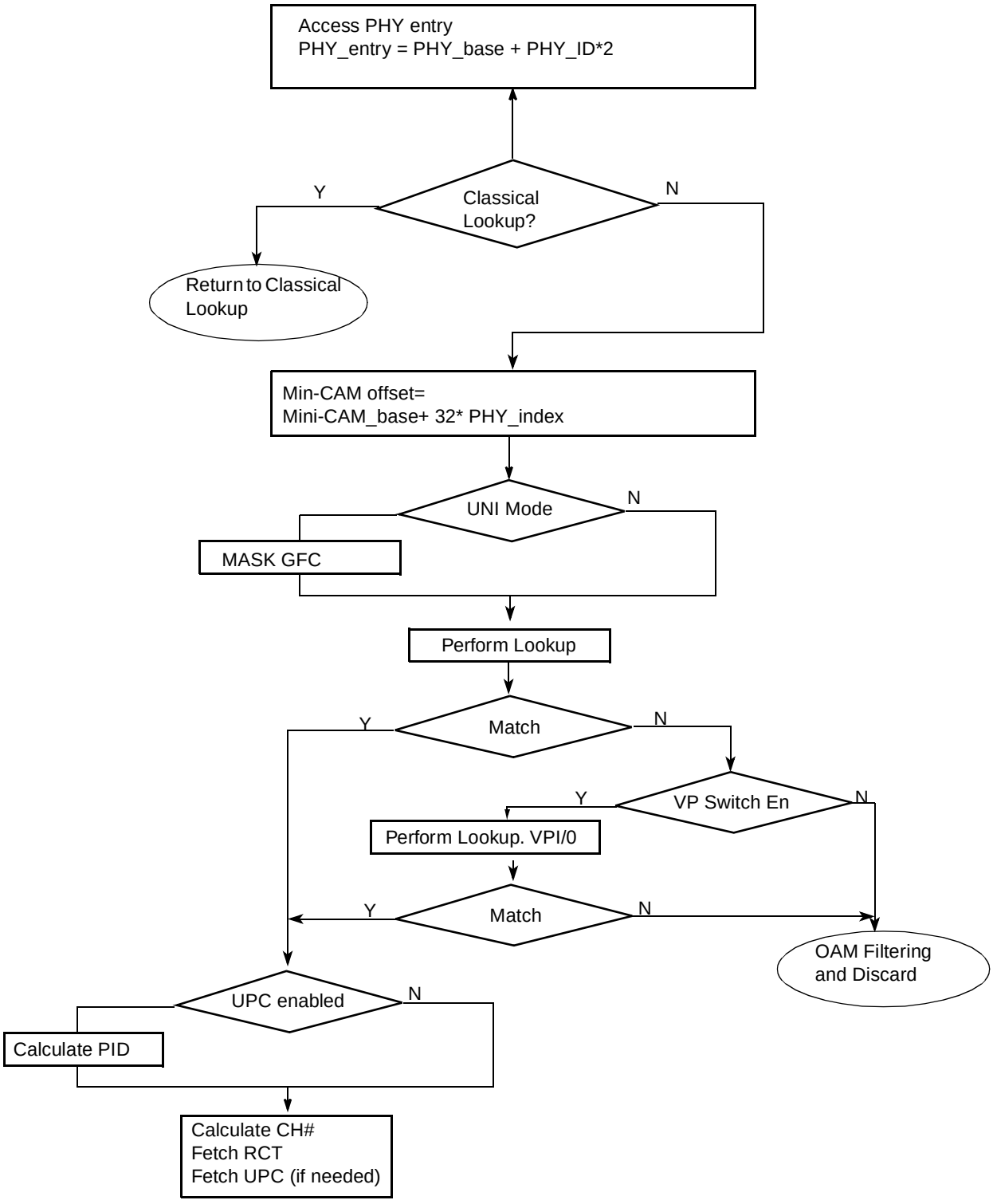


Figure 32-15. Mini-CAM Look-Up Flow

32.2.15 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 32.2.21, “ATM Layer Statistics.”

32.2.16 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 33.6.5.1, “UPDCx in ATM Protocol.” As shown in Figure 32-16, the extra header size can vary between 1 to 12 bytes (byte resolution) and the HEC octet is optional.

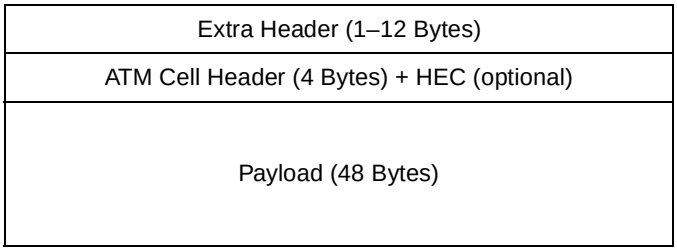


Figure 32-16. Format of User-Defined Cells

For AAL5 and AAL1 CES 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.

32.2.16.1 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 32.3.2.5, “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 32-6.

Table 32-6. 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

32.2.16.1.1 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 32.3.2.5, “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 32.2.15, “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).

32.2.17 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 32-18.

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 32-17 shows a flowchart of the ATM cell flow.

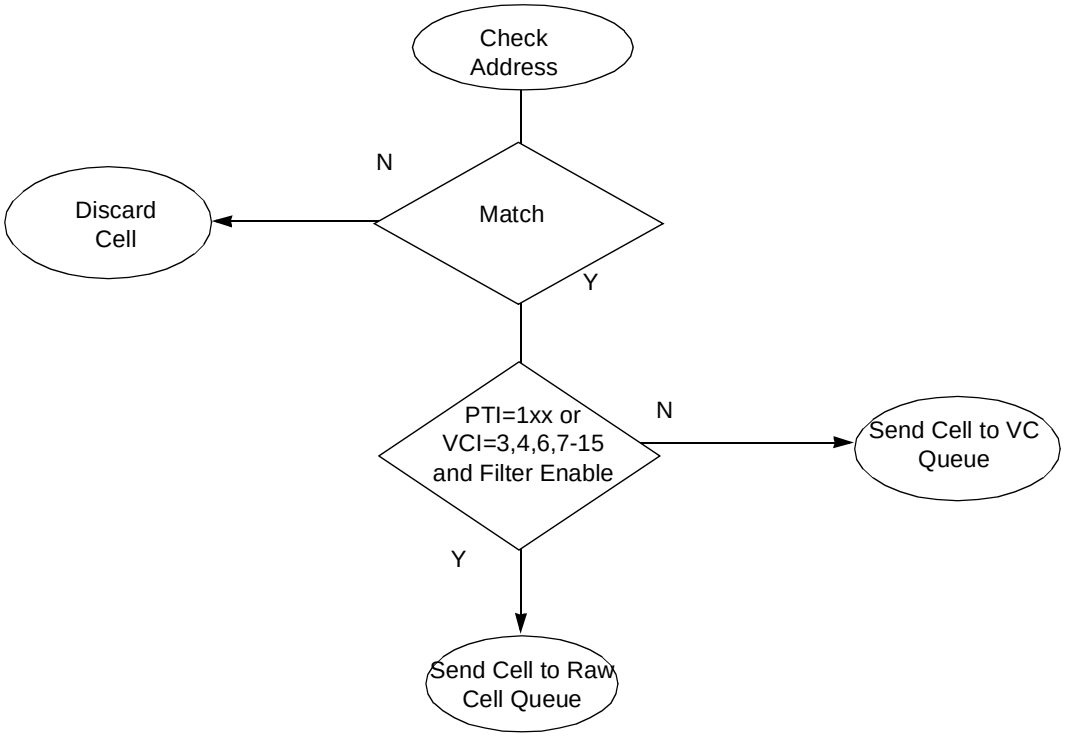


Figure 32-17. ATM Address Recognition Flowchart

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.

32.2.18 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.

32.2.18.1 ATM-Layer OAM Definitions

Table 32-7 lists pre-assigned header values at the user-network interface (UNI).

Table 32-7. 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 32-8 lists pre-assigned header values at the network-node interface (NNI).

Table 32-8. 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

32.2.18.2 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.

32.2.18.3 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.

32.2.18.4 Receiving OAM F4 or F5 Cells

OAM F4/F5 flow cells are received using the raw cell queue, described in Section 32.2.17, "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.

32.2.18.5 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 32.3.4.2.3, "AAL0 Protocol-Specific TCT." Prepare the OAM F4/F5 flow cell and insert it in an AAL0 TxBD. Finally, issue a ATM TRANSMIT command to send the OAM cell. For multiple PHYs, use several AAL0 channels—each PHY should have one transmit raw cell queue that is associated with its scheduling table.

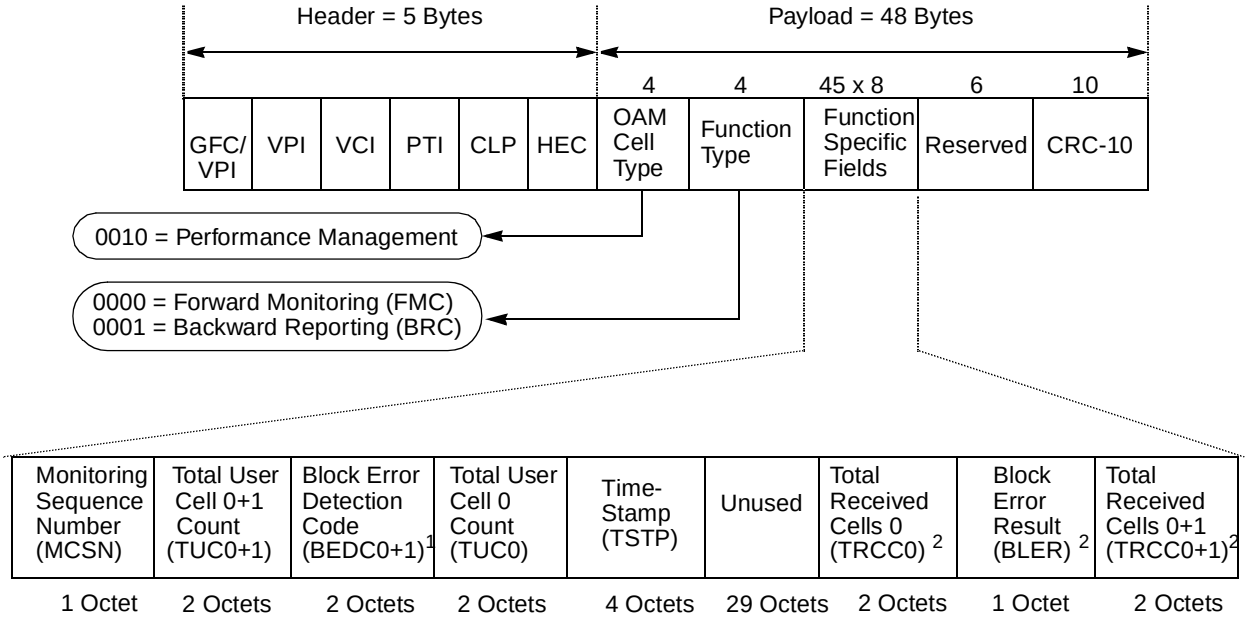
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel's TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are processed).

32.2.19 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 32-18 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 32-18. Performance Monitoring Cell Structure (FMCs and BRCs)

Table 32-9 describes performance monitoring cell fields.

Table 32-9. 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

32.2.19.1 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 32.3.5, "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.

32.2.19.2 PM Block Monitoring

PM block monitoring is done by the receiver. After initialization (see Section 32.2.19.1, "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.

32.2.19.3 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 20.3.9, "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 32.3.5, "OAM Performance Monitoring Tables."

In Figure 32-19, 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.

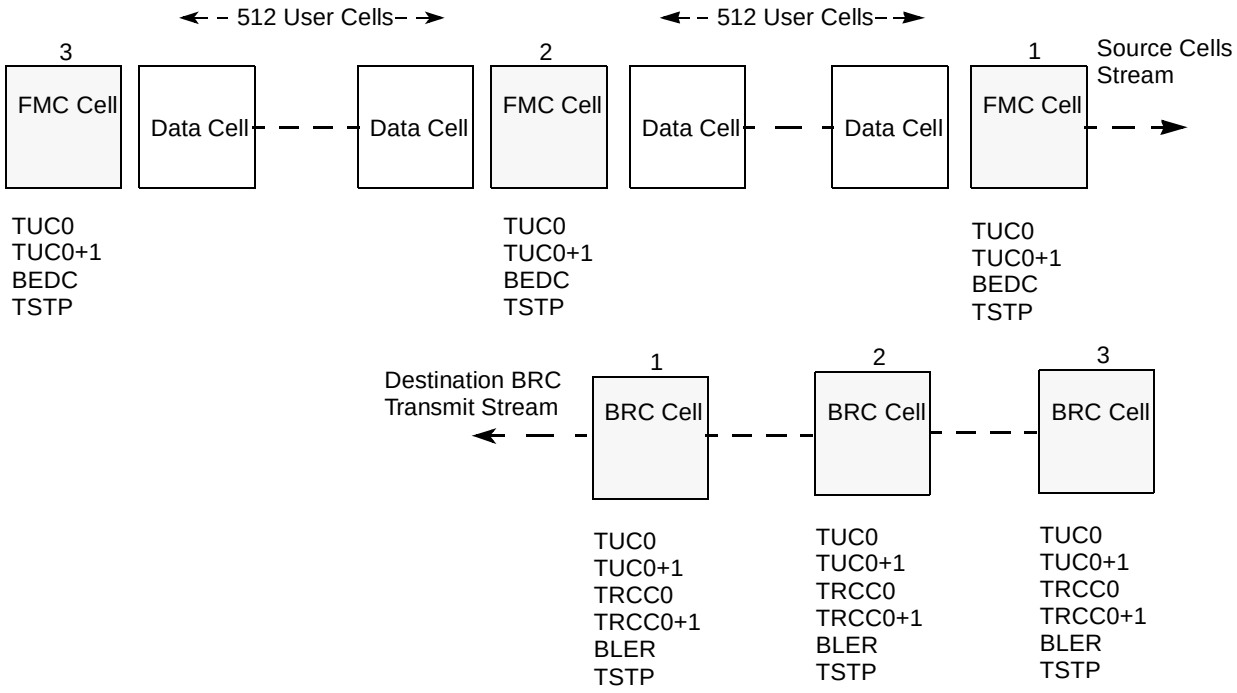


Figure 32-19. FMC, BRC Insertion

32.2.19.4 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.

32.2.20 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 32-5 and Table 32-4 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 20.3.9, "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 32-10. Both LBF1 and LBF2 have to be initialized by the user.

Table 32-10. 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.

32.2.20.1 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 32-64. 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 32-2. 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
```

32.2.20.2 UPC Operation Modes

32.2.20.2.1 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.

32.2.20.2.2 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.

32.2.20.2.3 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.

32.2.20.3 Examples of Dual Leaky Bucket Configurations

Some Leaky Bucket examples are shown in Table 32-11. The examples show Leaky Bucket programming for ATM Forum TM Specification 4.1 service Categories and ITU-T I.371 ATM Transfer Capabilities.

Table 32-11. 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

32.2.20.4 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).

32.2.21 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 32.3.12, "UNI Statistics Table."

32.2.22 ATM-to-TDM Interworking

The QUICC Engine block supports ATM and TDM Interworking. The MCC and the SI handle the TDM data processing. The ATM controller processes the ATM data.

Possible Interworking applications include the following:

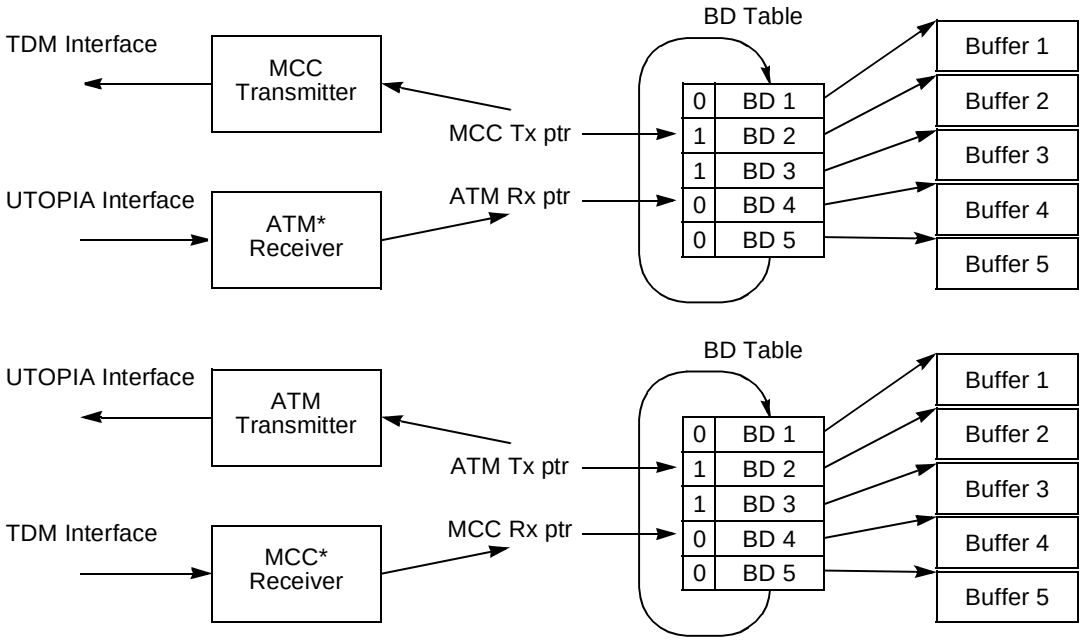
- Circuit emulation service (CES)
- Carrying voice over ATM
- Multiplexing several low speed services, such as voice and data, onto one ATM connection

Data forwarding between the ATM controller and an MCC can be done in two ways:

- Core intervention. When an MCC receive buffer is full and its RxBD is closed, the MCC interrupts the core. The core copies the MCC's receive buffer pointer to an ATM TxBD and sets the ready bit (TxBD[R]). Similarly, when an ATM receive buffer is full and its RxBD is closed, the core services the ATM controller's interrupt by copying the ATM receive buffer pointer to an MCC TxBD and setting TxBD[R]. This mode is useful when additional core processing is required.
- Automatic data forwarding. This mode enables automatic data forwarding between AAL1/AAL0 and transparent mode over a TDM interface.

32.2.22.1 Automatic Data Forwarding

The basic concept of automatic data forwarding is to program the ATM controller and the MCC to process the same BD table, as shown in Figure 32-20.



* The MCC and ATM receivers should be programmed to operate in opposite polarity E (empty) bit.

Figure 32-20. ATM-to-TDM Interworking

When going from TDM to ATM, the MCC receiver routes data from the TDM line to a specific BD table. The ATM controller transmitter is programmed to operate on the same table. When the MCC fills a receive buffer, the ATM controller sends it. The two controllers synchronize on the MCC's RxBD[E] and the ATM controller's TxBD[R].

When going from ATM to TDM, the ATM receiver reassembles data received from a particular channel to a specific BD table. The MCC transmitter is programmed to operate on the same table. When the ATM controller fills a receive buffer, the MCC controller sends it. The controllers synchronize on the ATM controller's RxBD[E] and the MCC's TxBD[R].

The MCC and ATM receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive data into buffers whose RxBD[E] = 0 and set RxBD[E] when a buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1- and AAL0-specific areas of the receive connection table, see Section 32.3.4.1, "Receive Connection Table (RCT)." For the MCC receiver, set CHAMR[EP], see Section 34.2.2.2.3, "Channel Mode Register (CHAMR)—Transparent Mode."

32.2.22.2 Using Interrupts in Automatic Data Forwarding

The core can program the MCC and ATM interrupt mechanism to trigger interrupts for events such as a buffer closing or transfer errors. The interrupt mechanism can be used to synchronize the start of the automatic bridging process. For example, to start the MCC transmitter after a specific buffer reaches the ATM receiver (the buffering is required to cope with the ATM network's CDV), set ATM RxBD[I]. When the receive buffer is full, the RxBD is closed, RxBD[E] is set (because it is operating in opposite E-bit polarity), and the core is interrupted. The core then starts the MCC transmitter.

32.2.22.3 Timing Issues

Use of the TDM interface assumes that all communicating entities are synchronized (that is, that they are using a synchronized serial clock). If the TDM interfaces are not synchronized, a slip can occur in the reassembly buffer. If a buffer-not-ready event occurs at the MCC transmitter, the user must restart the MCC transmit channel.

32.2.22.4 Clock Synchronization (SRTS and Adaptive FIFOs)

Clock synchronization methods, such as using a time stamp (SRTS) or adaptive FIFOs, prevent buffer slipping during reassembly. The SRTS method may be implemented using external logic. The QUICC Engine block can read the SRTS from external logic and insert it into AAL1 CES cells, and can track the SRTS from AAL1 CES cells and deliver it to external logic. See Section 32.2.22.4.1, “SRTS Generation and Clock Recovery Using External Logic.”

Alternatively, an adaptive FIFOs method can be implemented using the core to maintain the bridging buffer at a mid-level point. The difference between the MCC and ATM data pointers is a measure of buffer synchronization. The core calculates the difference between pointers at regular intervals and adapts the TDM clock accordingly to hold the difference constant.

32.2.22.4.1 SRTS Generation and Clock Recovery Using External Logic

The QUICC Engine block supports SRTS generation using external logic. If SRTS generation is enabled (TCT[SRT] = 1), the QUICC Engine block reads SRTS[0–3] from the external SRTS logic and inserts it into 4 cells whose SN fields equal 1, 3, 5, and 7, as shown in Figure 32-21.

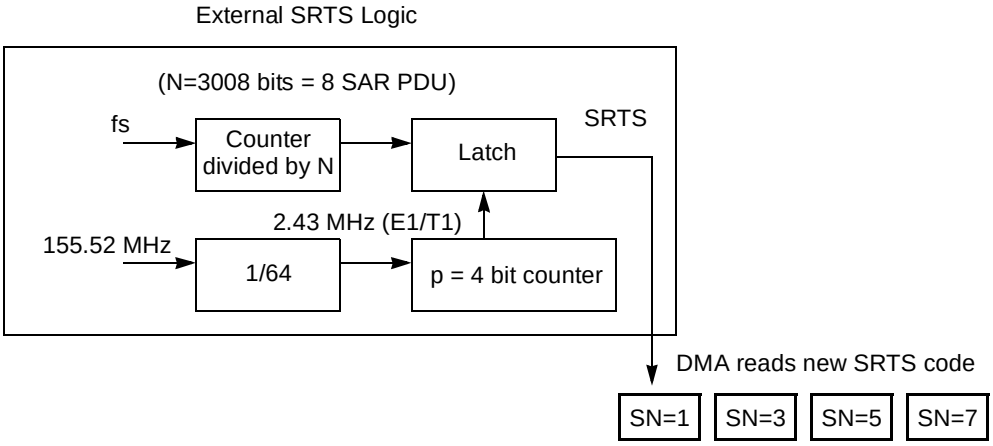


Figure 32-21. AAL1 CES SRTS Generation Using External Logic

For every eight cells, the external SRTS logic should supply a valid SRTS code. The QUICC Engine block reads the SRTS code from the bus selected in TCT[BDB] using a DMA read cycle of 1-byte data size. Each AAL1 CES channel can be programmed to select one of 16 addresses available for reading the SRTS result. The SRTS code should be placed on the least-significant nibble of that address (SRTS[0]=lsb, SRTS[3]=msb). The SRTS is synchronized with the sequence count cycle—SRTS[0] is inserted into the cell with SN = 7; SRTS[3] is inserted into the cell with SN = 1. For every eighth AAL1 CES SAR PDU,

the SRTS logic samples a new SRTS and stores it internally. The SRTS is a sample of a 4-bit counter with a 2.43-MHz reference clock (for E1/T1) synchronized with the network clock.

The QUICC Engine block supports clock recovery using an external SRTS PLL. If SRTS recovery is enabled (RCT[SRT]=1), the QUICC Engine block tracks the SRTS from four incoming cells whose SN field equals 1, 3, 5, and 7 and writes the result to external SRTS logic, as shown in Figure 32-22.

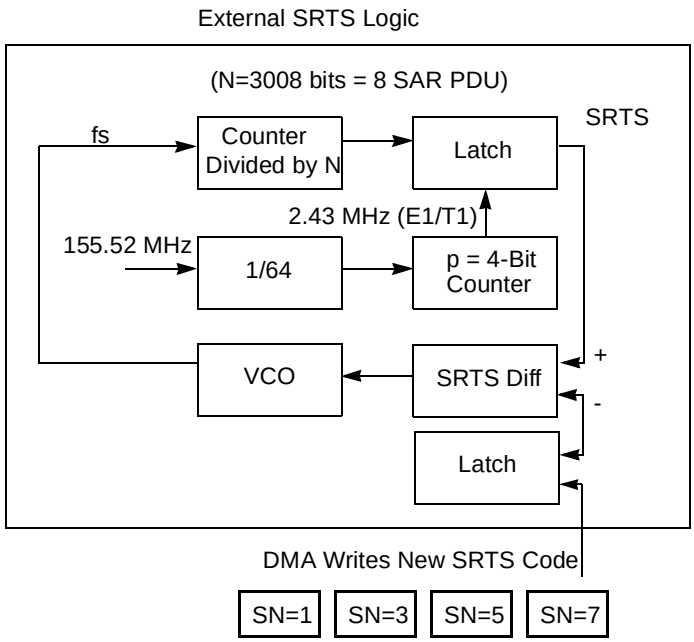


Figure 32-22. AAL1 CES SRTS Clock Recovery Using External Logic

On every eighth cell, the QUICC Engine block writes a new SRTS code to the external logic using the bus selected in RCT[BDB]. The QUICC Engine block writes the SRTS code using a DMA write cycle of 1-byte data size. Each AAL1 CES channel can be programmed to select one of 16 addresses available for writing the SRTS result. The SRTS code is written to the least-significant nibble of that address (SRTS[0]=lsb, SRTS[3]=msb). The SRTS is synchronized with the sequence count cycle—SRTS[3] is read from the cell with SN = 1 and SRTS[0] is read from the cell with SN = 7. The SRTS PLL makes periodic clock adjustments based on the difference between a locally generated SRTS and a remotely generated SRTS retrieved every eight received cells.

32.2.22.5 Mapping TDM Time Slots to VCs

Using the MCC and the SI, any TDM time-slot combination can be routed to a specific data buffer, see Section 34.2.5, “MCC Configuration Registers (MCCF) and Chapter 19, “System Interface.” The same data buffers should be used by the ATM controller to route receive and transmit data. For information about ATM buffers see Section 32.3.9, “ATM Controller Buffer Descriptors (BDs).”

32.2.22.6 CAS Support

For applications requiring channel-associated signaling (CAS), circuit emulation with CAS requires additional core processing. External framers perform the CAS manipulation through a serial or parallel interface.

When the MCC receives a multi-frame block, it generates an interrupt to the core. The core reads the CAS block from the external framer and places it at the end of the ATM data buffer after the structured multi-frame block. The core then passes the buffer pointer to the ATM controller, and the controller packs the data and CAS block into AAL1 CES cells. All AAL1 CES functions, such as generating PDU-headers and structured pointers, operate normally.

When the ATM controller receives a multi-frame block, it generates an interrupt to the core. The core reads the CAS block from the data buffer and writes it to the external framer. The core then moves the buffer pointer to the MCC. The buffer's data length should not include the CAS octets.

To optimize the process, the framer may interrupt the core only when the CAS information changes. (CAS information changes slowly.) The core can keep the CAS block in memory and connect to the framer only when the CAS changes. The core can use regular read and write cycles when connecting to the framer through a parallel interface.

The MCC and ATM controller should be synchronized with the framer's multi-frame block boundary. At the ATM side, the structured block size should equal the multi-frame block size plus the size of the CAS block so that the structured pointer, inserted by the ATM controller, points to the start of the structured data block. At the MCC side, the MCC must to be synchronized with the super frame sync signal. This synchronization can be achieved by external logic that triggers on the super frame sync signal and starts delivering the frame sync to the MCC. When loss of super frame synchronization occurs, this logic should reset and trigger again on the next super frame indication.

32.2.22.7 Trunk Condition

According to the Bellcore standard, the Interworking function (IWF) should be able to transmit special payload on both ATM and TDM channels to signal alarm conditions (Bellcore TR-NWT-000170). The core can be used to generate the trunk condition payload in special buffers (or existing buffers) for the ATM controller or MCC.

32.2.22.8 ATM-to-ATM Data Forwarding

A simple automatic data forwarding can be used to switch ATM AAL0 cells from one ATM port to another without core intervention. The ATM receiver and transmitter should be programed to process the same BD table. When the ATM receiver fills an AAL0 buffer, the ATM transmitter sends it. The ATM receiver and transmitter are synchronized using the same mechanism as described for ATM-to-TDM automatic forwarding; see Section 32.2.22.1, "Automatic Data Forwarding."

A more complex and feature reach method of ATM switching is available as MSP- Multi-Service Platform which is described in Chapter 39, "Enhanced MSP Microcode."

32.3 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 AAL1 CES sequence number protection table, the UNI statistics table and the UPC table.

32.3.1 Parameter RAM Page Organization

Figure 32-23 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.

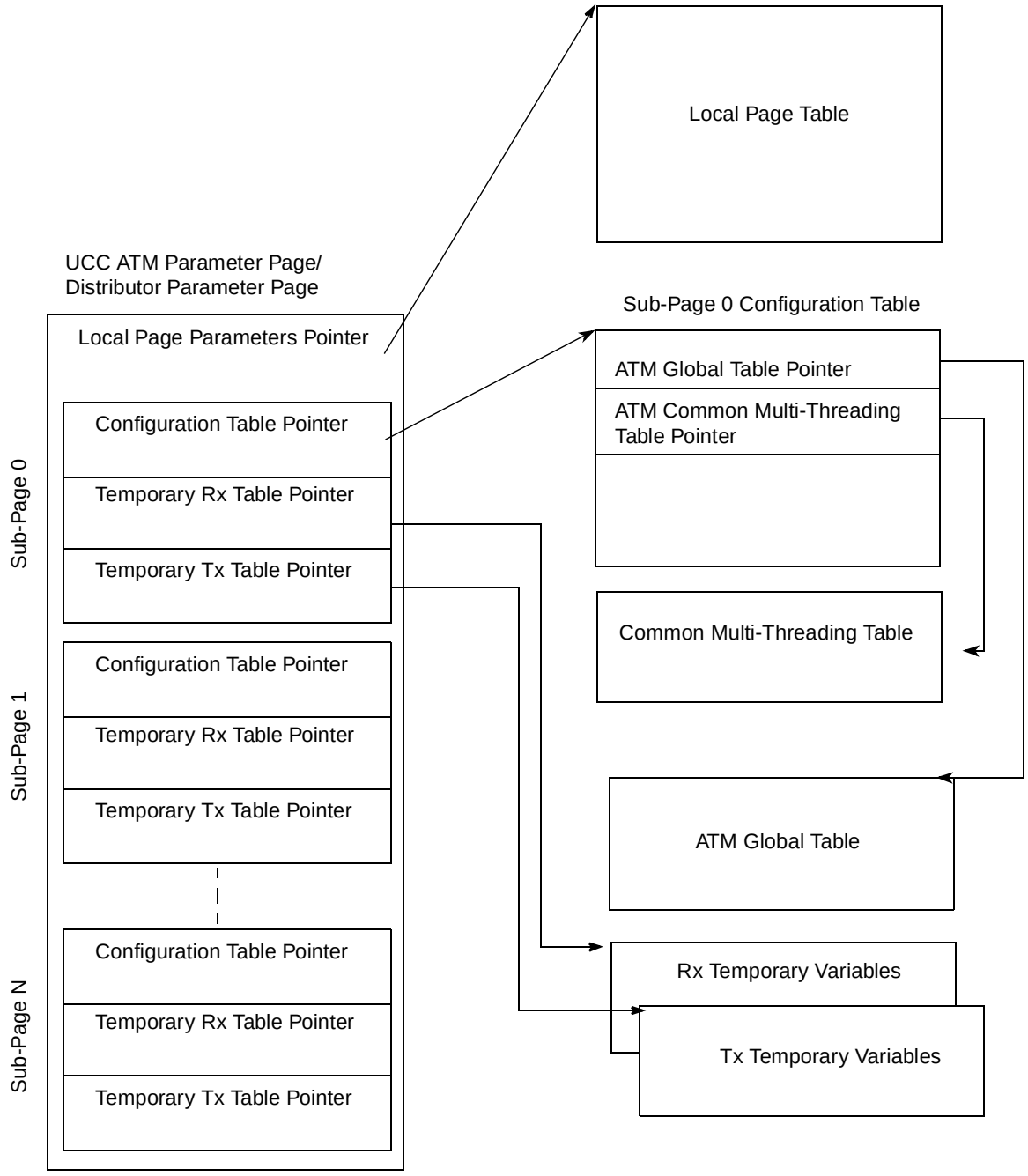


Figure 32-23. Parameter Page Organization

32.3.2 Parameter RAM

When configured for ATM mode, the UCC parameter page is mapped as shown in Table 32-12. 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.

32.3.2.1 Parameter RAM for Non Multi-Threading Operation

Table 32-12 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 32.3.2.2, "Parameter RAM for Multi-threading Operation."

Table 32-12. UCC Parameter RAM Page

Offset	Size	Field	Description	Mem 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 32-13. **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, AAL2 and AAL1- CES. The address is 8 bytes aligned.	Allocate 0x60 bytes. see Table 32-18 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 E2AAL2, Policer or MSP are 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 E2AAL2, Policer or MSP are 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.

Table 32-12. UCC Parameter RAM Page (continued)

Offset	Size	Field	Description	Mem Allocation
0x0c	HW	**sub-page1_Tx_ Tmp_Table_ptr**	**Initialize when E2AAL2, Policer or MSP are 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 32-24 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.

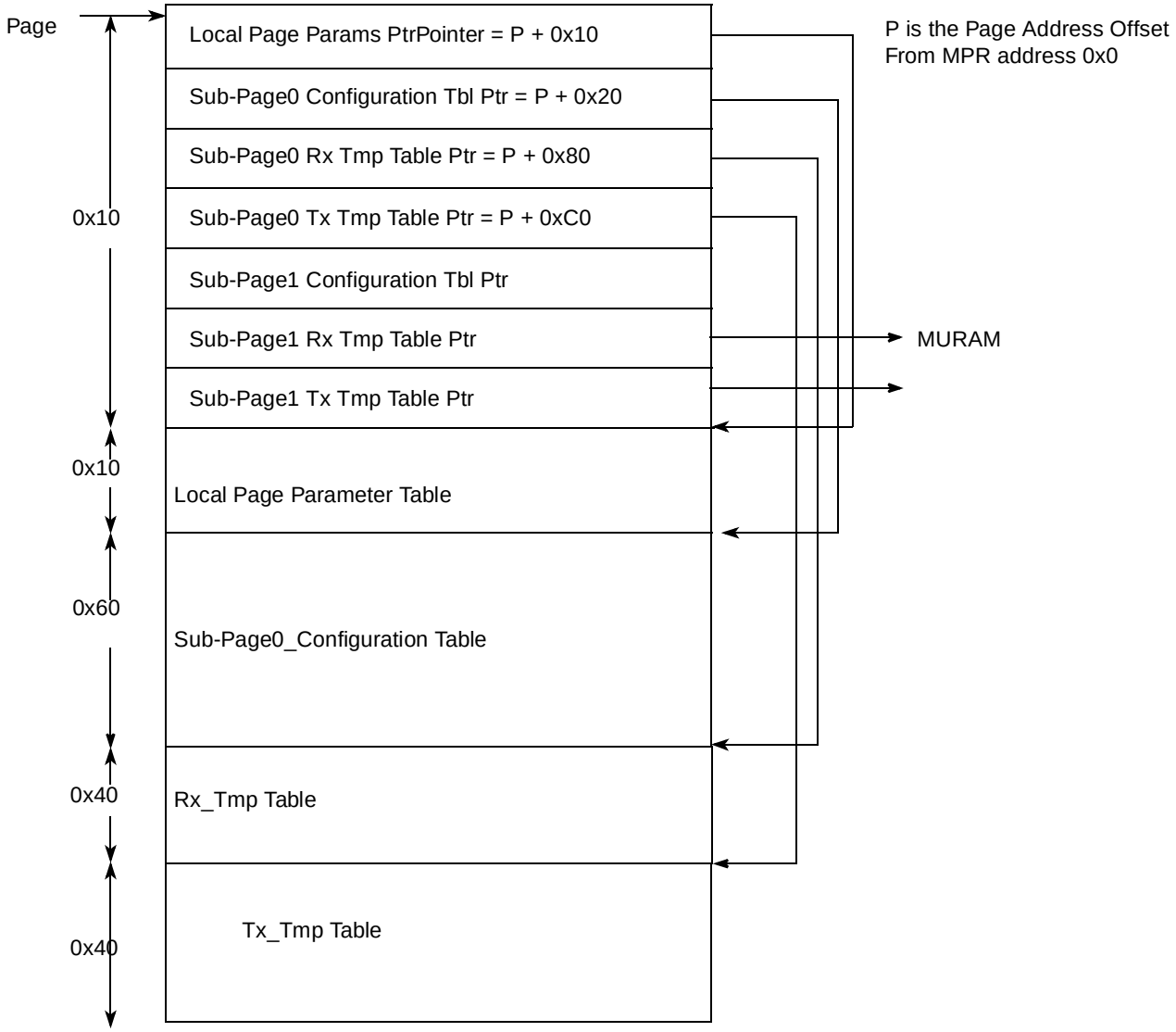


Figure 32-24. Possible Configuration for 0x100 Bytes UCC Page

Table 32-13 is pointed from distributor page see Table 32-12 [Local_page_parameters_ptr]. For Non-Multi-threading mode the size of this table is reduced to 0x10 bytes.

Table 32-13. UCC Local Page Parameter Table

Offset	Size	Field	Description	Non Multi-Threading Mem Allocation
0x00	HW	**INT_RCT_TMP_ptr**	Memory location to where an external RCT is dma'ed.	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	**RxQD_tmp** **AAL1_INT_RX_CRT**	Dual purpose. AAL2: Memory location to where an AAL2 RxQD is dma'ed CES: Memory location to where the CAS Rx Routing table is fetched	If AAL2 or CES is enabled allocate 0x20 bytes. 32 bytes aligned
0x08	HW	**PPRS_INT_PTR/ AAL1_INT_TX_CRT**	Dual purpose. 1. AAL2:Partial Packet Receive Storage internal pointer. Valid only if PPRS mode in enabled in the RCT. Allocates 64 bytes in RAM, needed for the QUICC Engine block in order to fetch the partial packet from PPRS external table. 2. CES: Memory location to where the CAS Tx Routing table is fetched	Allocate 0x40 bytes for temporary storage of partial packet or CAS routing table.
0x0A	HW	**Res**	Reserved- Should be cleared	
0x0C	W	**Res**	Reserved- Should be cleared	

32.3.2.2 Parameter RAM for Multi-threading Operation

Table 32-14 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 32.3.2.1, "Parameter RAM for Non Multi-Threading Operation."

Table 32-14. Distributor Parameter RAM Page

Offset	Size	Field	Description	Multi-Threading Mem 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 32-15

Table 32-14. Distributor Parameter RAM Page (continued)

Offset	Size	Field	Description	Multi-Threading Mem Allocation
0x02	HW	**sub-page0_ Configuration_ Table_ptr**	This is a pointer to static configuration parameters table. Support for AAL0, AAL5, AAL2 and AAL1- CES. The address is 8 bytes aligned.	Allocate 0x80 bytes. See Table 32-18 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.
0x08	HW	**sub-page1_ Configuration_ Table_ptr**	**Initialize when E2AAL2, Policer or MSP are enabled.** This is a pointer to static configuration parameters table. Support for E2AAL2, Policer and MSP. 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 32-15 is pointed from distributor page see Table 32-14 [Local_page_parameters_ptr]. For Multi-Threading mode the table has to be fully initialized.

Table 32-15. Distributor Local Page Parameter Table

Offset	Size	Field	Description	Multi-Threading Mem 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 32-16 describes the Thread parameter table which has to be assigned and initialized for each of the

ATM threads available in the system.

Table 32-16. 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 32-17 for initialization details.
0x02	HW	**sub-page0_Configuration_Table_ptr**	This is a pointer to static configuration parameters table. Support for AAL0, AAL5, AAL2 and AAL1- CES. 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.
0x08	HW	**sub-page1_Configuration_Table_ptr**	This is a pointer to static configuration parameters table. Support for E2AAL2, Policer and MSP. 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.

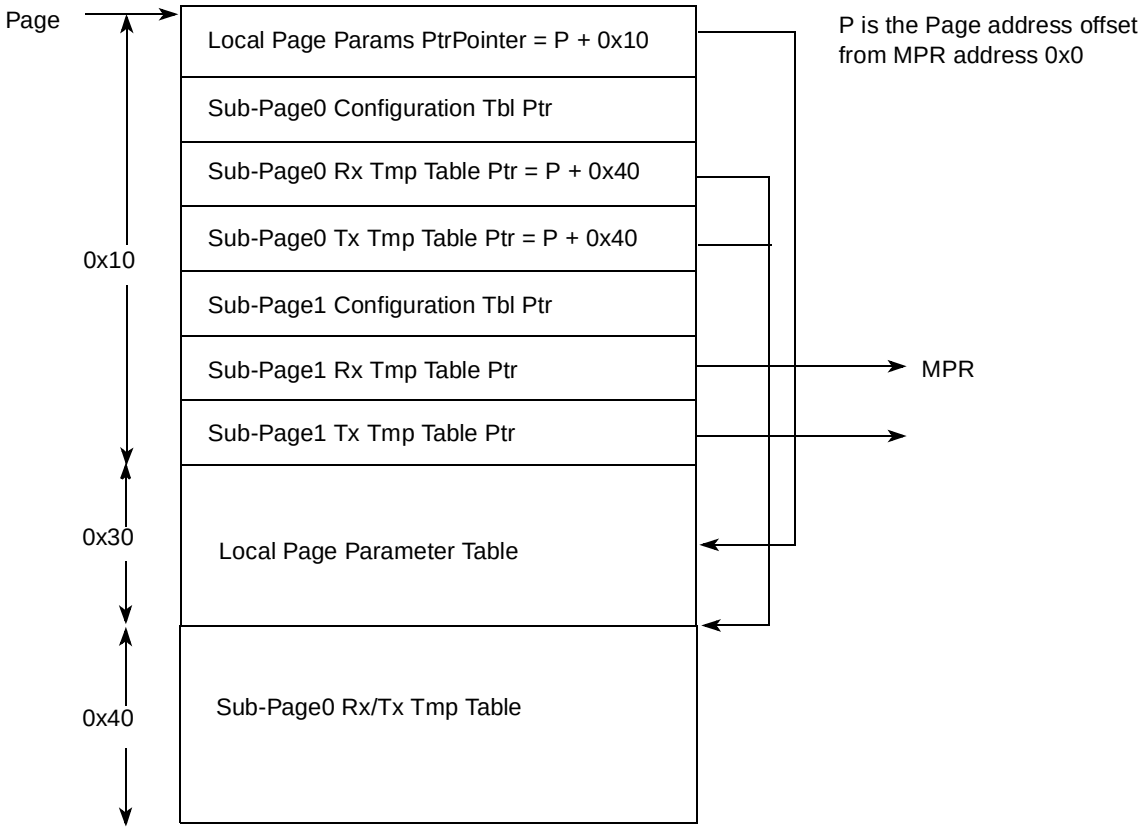


Figure 32-25. Possible Configuration for a Thread Page

Figure 32-25 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 32-17. Thread Local Page Parameter Table

Offset	Size	Field	Description	Mem Allocation
0x00	HW	**INT_RCT_TMP_ptr**	Memory location to where an external RCT is dma'ed	Allocate 0x20 bytes. 32 bytes aligned
0x02	HW	**Res**	reserved should be cleared.	—
0x04	HW	**Res**	reserved should be cleared.	—
0x06	HW	**RxQD_tmp** **Rx_Des_Base_Int/** **AAL1_INT_RX_CRT**	Dual purpose. AAL2: Memory location to where an AAL2 RxQD is dma'ed CES: Memory location to where the CAS Rx Routing table is fetched	If AAL2 or CES is enabled allocate 0x20 bytes. 32 bytes aligned

Table 32-17. Thread Local Page Parameter Table (continued)

Offset	Size	Field	Description	Mem Allocation
0x08	HW	**PPRS_INT_PTR/ AAL1_INT_TX_CRT**	Dual purpose. 1. AAL2:Partial Packet Receive Storage internal pointer. Valid only if PPRS mode in enabled in the RCT. Allocates 64 bytes in RAM, needed for the QUICC Engine block in order to fetch the partial packet from PPRS external table. 2. CES: Memory location to where the CAS Tx Routing table is fetched	Allocate 0x40 bytes for temporary storage of partial packet or CAS routing table.
0x0A–0x2F	Res	**Reserved**	Reserved should be cleared	—

Table 32-18 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, AAL1 and AAL2. This table is pointed from distributor page Table 32-12 [sub-page0_Configuration_Table_ptr].

Table 32-18. Sub-page0 Configuration Table

Offset	Size	Field	Description	Mem Allocation
0x00	HW	**Global ATM Parameters Table ptr**	A pointer to a Common ATM parameters for all the UCCs see Table 32-19 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 32.3.2.6/32-72.	
0x08	W	**Reserved**	Reserved. Should be cleared.	
0x0c	HW	**GMODE**	Global mode. User-defined. See 32.3.2.5/32-70.	
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 32-26.	points to a 12 bytes table (if address compression is enabled).

Table 32-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Mem 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 32-27.	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 32.3.2.4/32-70.	
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 32.3.12/32-136.	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 32.3.2.3/32-69.	

Table 32-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Mem Allocation
0x22	HW	**IDLE_BASE**	Valid only when Serial ATM 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. 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	**Res**	Reserved Should be cleared	
0x28	W	**TCELL_TMP_BASE_EXT**	Used for AAL1 CES and AAL2. For AAL2 operation with count CU enabled. Temporary location base address for storing partially filled ATM AAL2 cells which are still awaiting the count to expire or the full payload. Points to a total of 64*last_AAL2_Ch# octets reserved in external memory for the partially filled cells. For AAL1 CES this is a temporary location for storing partially filled cells when there is no data available for transmission in the next BD. Bus location is identical to the bus on which the data resides (imposes same bus usage for all the AAL2 channels using this feature).	64bytes * number of AAL2 channels.
0x2C	HW	**RxQD_Base_Int**	Points to the base address of the internal RxQD table. The pointer should be 32-byte aligned. User-defined.	
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	**AAL1_Ext_STATS_BASE**	AAL1 External Statistics Table Base. Points to External memory. Should be 16-byte aligned (16 octets for each AAL1 channel).User-defined. See Section 35.16, “External AAL1 CES Statistics Tables.”	

Table 32-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Mem Allocation
0x34	HW	**AAL1_DUMMY_CELL_BASE**	ALL1 Dummy cell base address. Points to multi-user RAM area contains the AAL1 dummy cell template (little-endian format). Should be 64-byte aligned. User-defined.	
0x36	HW	**INT_RTCRT_BASE**	Internal receive/transmit CAS routing table extension base. User-defined. Should be 64-byte aligned. Each transmitter entry is accessed using: INT_RTCRT_BASE + cc * 64 and for a receiver INT_RTCRT_BASE + CC * 64 + 32.Note that because AAL1 CES does not need the TCT extension, the AAL1 CES microcode uses this Hword to point to a CAS routing table. Should be 32 byte aligned. User-defined.	
0x38	W	**EXT_RTCRT_BASE**	External receive/transmit CAS routing table extension base. User-defined. Should be 64-byte aligned. Each transmitter entry is accessed using: EXT_RTCRT_BASE + CC * 64 and for a receiver EXT_RTCRT_BASE + CC * 64 + 32.Note that because AAL1 CES does not need the TCT extension, the AAL1 CES microcode uses this word to point to a CAS routing table.	
0x3c	HW	**AAL1_SNPT_BASE**	AAL1 SNP protection look-up table base address. (AAL1 and CES only.) The 32-byte table resides in multi-user RAM. AAL1_SNPT_BASE must be halfword-aligned. User-defined offset from multi-user RAM base. See 32.3.11/32-136.	
0x3e	HW	**ADAPTIVE_THRESHOLDS_BASE (CATB)**	CES adaptive threshold tables base address. Points to the multi-user RAM area containing the CES slip control thresholds and the adaptive counter See Section 35.15, “Internal AAL1 CES Statistics Tables.” Should be 8-byte aligned (8 octets for each AAL1-MCC channel). User-defined and should match the CATB value programmed in the MCC parameter RAM; see Section 34.2.2.3, “MCC Parameters for AAL1 CES Usage.”	

Table 32-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Mem Allocation
0x40	W	**SRTS_BASE**	External SRTS logic base address. AAL1 and CES only. Should be 16-byte aligned. The four least-significant bits are taken from SRTS_DEV in the AAL1-specific area of the connection table entries.	
0x44	HW	**IN_CAS_BLOCK_BASE**	Incoming CAS Block Base (depicted in Figure 35-12). Points to multi-user RAM. Should be 32-byte aligned. User-defined.	
0x46	HW	**OUT_CAS_BLOCK_BASE**	Outgoing CAS Block Base (depicted in Figure 32-66). Points to multi-user RAM. Should be 32-byte aligned. User-defined.	
0x48	HW	**AAL1_Out_CAS_Status_Reg (OCASSR)**	Outgoing CAS Status Register. See Section 35.10, “Outgoing CAS Status Register (OCASSR).”	
0x4a	HW	**Res**	Reserved. Should be cleared	
0x4c	HW	**AAL1_Int_STATS_BASE**	AAL1 Internal Statistics Table Base. Points to multi-user RAM. Should be 16-byte aligned. User-defined. See Section 35.15, “Internal AAL1 CES Statistics Tables.”	
0x4e	HW	**EAAL2_PAD_TMP_BASE**	PAD template base address. Points to an internal memory area that contains the zero cell template. Should be 64-byte aligned. User-defined.	64 bytes
0x50	W	**RAS_Timer_Duration**	Contains the RAS_Timer duration in time units as defined by Time Stamp1 Register (CETSCR1) for the SSSAR sublayer. User-defined.	
0x54	W	**RxQD_Base_Ext**	Points to the base address of the external RxQD table. The actual address of the first RxQD in the table is RxQD_Base_Ext + 512*4. User-defined.	

Table 32-18. Sub-page0 Configuration Table (continued)

Offset	Size	Field	Description	Mem Allocation
0x58	W	**Rx_UDC_BASE**	Valid only for AAL2 VCs. Points to the base of the RX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: RX_UDC_Base + 16*CH# (where CH# is the ATM channel number) Bus location is identical to the bus on which the data resides (imposes same bus usage for all channels)	
0x5c	W	**Tx_UDC_BASE**	Valid only for AAL2 VCs. Points to the base of the TX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: TX_UDC_Base + 16*CH# (where CH# is the ATM channel number) Bus location is identical to the bus on which the data resides (imposes same bus usage for all channels)	
0x60	Byte	**Rx Terminator SNUM**	The SNUM which is used for the receiver terminator process. For available thread numbers see Table 20-15. 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 20-15. 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 32-20.	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 32-26.	
0x6C	W	**ATM_Available_Tx_Thread_MASK**	32 bit indicating available threads for this transmitter UCC. See Figure 32-26.	

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.

32.3.2.2.1 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 32.2.3, “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 it’s 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.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Th0	Th1	Th2	Th3	Th4	Th5	Th6	Th7	Th8	Th9	Th10	Th11	Th12	Th13	Th14	Th15
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	Th16	Th17	Th18	Th19	Th20	Th21	Th22	Th23	Th24	Th25	Th26	Th27	Th28	Th29	Th30	Th31

Figure 32-26. ATM_Available_Xx_Thread_Mask

Table 32-19 shows the Global ATM parameters Table. It is pointed from Table 32-18 [Global ATM Parameters Table ptr].

Table 32-19. 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.

Table 32-19. Global ATM Parameters Table (continued)

Offset	Size	Field	Description
0x12	HW	**COMM_INFO_CTL**	The information field associated with the last host command. User-defined. See 32.4.1/32-142.
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	**ATM_AVCON_BASE**	ATM Auto VC ON Table Base. Valid for ATM switch mode, when the TCT[AVCF] is set and when the TCT[VCON] bit is cleared. The AVCON mechanism **must** be enabled on the receiver side, see for AAL2 mode the RxQD[AVCON] bit (Section 41.4.3, “AAL2 Switching”). This is a Pointer to a multi port RAM array, which contains one bit per TCT channel. see Figure 32-27. If the switched channel is scheduled using the GCRA scheduler, DRR APC the VCON bit is located at the GCRA scheduler, DRR APC structure and is not required in the array. Used in MSP in hierarchical APC mode .
0x20–0x30	16 Bytes	**Res**	Reserved. Should be cleared.

Figure 32-27 describes the ATM AVCON array. For better memory utilization it is recommended that the switched TCTs will be consecutive channels, to eliminate gaps in this table. The area dedicated for this table is equal to the integer “round up” number of switch TCTs (which participate in the AVCON/AVCF mechanism) divided by 8. This area should be initialized with zero.

The transmitter clears the corresponding TCT bit on event of deactivating a channel, and the receiver, when detecting that this bit is cleared, will activate the channel and set this bit.

Figure 32-27. ATM AVCON Array

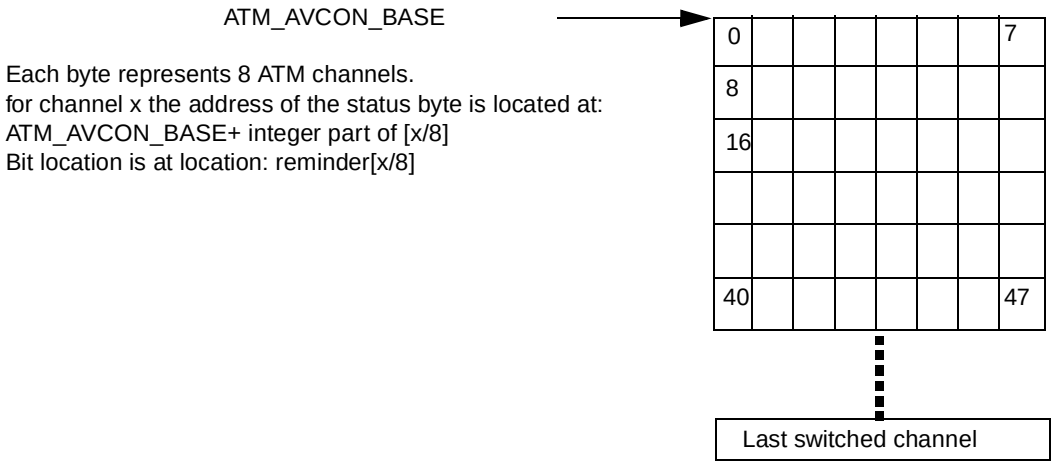


Table 32-20 contains the Common MTH Parameters. It is pointed from Table 32-18 [Common MTH parameters Table Base]. The table contains all the management parameters for the Multi-threaded

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 32-20. Common MTH Parameters Table

Offset	Size	Field	Description	Mem 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 32-30 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	**ATM_THREAD_CAM_SIZE**	The size of the (ATM_thread_CAM table in bytes-1). equal to (Number of threads* 4 -1).	
0x09–20				

Table 32-21 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 AAL2, policer and MSP operation.This table is pointed from Table 32-12 [sub-page1_Configuration_Table_ptr].

Table 32-21. Sub-page1 Configuration Table

Offset	Size	Field	Description
0x00	W	**TxQDExtBase**	E2AAL2 Mode: TxQD external RAM Base address. All external TxQDs are located as offsets from this field. Must be 32 bytes aligned. See Section 41.3.5.5, "SSSAR Tx Queue Descriptor."

Table 32-21. Sub-page1 Configuration Table (continued)

Offset	Size	Field	Description
0x04	W	**TxCIDStatsAddrBASE**	E2AAL2 Mode: External RAM TxCIDStatsAddr Table Base address, used when TCT[ExtStatsAddr] bit is set. Must be Word aligned. See Section 41.3.5.1, "AAL2 Transmit Connection Tables."
0x08	HW	**TxCIDStatsOffsetBASE**	E2AAL2 Mode: Internal RAM TxCIDStatsOffset Table Base address, used when TCT[ExtStatsAddr] bit is cleared. Must be HWord aligned. See Section 41.3.5.1, "AAL2 Transmit Connection Tables."
0x0A	HW	**Res**	Reserved. Should be cleared.
0x0C	W	**PPRS_EXT_BASE**	E2AAL2 Mode: Partial Packet Receive Storage External Base. When PPRS mode is enabled in the RCT, the user should allocate a table in external memory of 64 bytes multiplied by the number of RCT channels. Each channel store an incoming partial packet at an address which is PPRS_EXT_BASE + 64* Ch# Should be 64bytes aligned. IMPOTRANT: The bus selection for this area is determined by the LSB bit of this entry (overriding the alignment requirements) If the LSB= 0 it resides on the coherent system bus. If the LSB=1 it resides on the QUICC Engine secondary bus.
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.
0x16	HW	**FS_BASE**	MSP Mode: FIFO Status Table Base Address in multi-user RAM. The FIFO Status table contains the FIFO Status Entries: One byte entry for each Hierarchical connection. The value of FS_BASE should be the same for all the UCCs.
0x18	HW	**QLTS_BASE**	MSP Mode: Queue Level Threshold Set table base address. User-defined offset from multi-user RAM base. The value of QLTS_BASE must be 16 byte aligned (three least significant bits are zero).
0x1A	HW	**FDTP_BASE**	MSP Mode: FIFO Descriptor Pointers Table Base. Should be aligned to 8 bytes. See Section 39.18.2, "FIFO Descriptor Pointers Table."
0x1C	HW	**FCP_BASE**	MSP Mode: Free Cell Pool Descriptor base address. User-defined offset from multi-user RAM base. If one free cell pool is used, 32 bytes need to be allocated in the DPR. If two free cell Pools are used 64 bytes need to be allocated.

Table 32-21. Sub-page1 Configuration Table (continued)

Offset	Size	Field	Description
0x1E	HW	**INS_CELL_TMP_PTR**	MSP Mode: Insert Cell Temporary pointer field. Used only for insert of cell command for FTCT/HTCT modes. Pointer to multi-user RAM 8 byte location (should be 8 bytes aligned), which are used by the QUICC Engine block to build the COV of the cell that would be written to the FIFO, for the “dummy” receive of the insert cell command.
0x20	HW	**MCE_BASE**	MSP Mode: Pointer for multi-cast event mask image table base address. Should be defined for the UCC where the MRCT resides.
0x22	HW	**FEPTR_BASE**	MSP Mode: Pointer to table per MRCT[MCGID], which is used to point to another table of WFTs pointers, for Multicast-FIFO with auto FE mechanism. Should be defined for the UCC where the MRCT resides.
0x24	W	**MSP_UPC_BASE**	MSP Mode: External memory Pointer to UPC parameter table Base address. Should be aligned to 32 bytes.
0x28	W	**EXT_INS_CELL_BASE**	MSP Mode: External Insert Cell BASE. Should be aligned to 8 bytes. See Section 39.15.3, “Insert Cell Mode.”
0x2C	HW	**INT_INS_CELL_BASE**	MSP Mode: Internal Insert Cell BASE. The user should always allocate at least one entry at INT_INS_CELL_BASE (associated with channel 0). Should be aligned to 8 bytes. See Section 39.15.3, “Insert Cell Mode.”
0x2E	Byte	**INS_CELL_Header_LOC**	MSP Mode: Insert Cell header Location. 0x0000 - Header is located in the Insert cell template (each table entry consist of 4bytes). 0x8000 - Header is located in the Insert Cell Table (each table entry consist of 8 bytes).
0x2F	Byte	**Res**	Reserved. Should be cleared
0x30	HW	**AAL2_THresholds_Tbl_Base**	E2AAL2 Mode: Pointer to an internal table where the queue management thresholds sets are stored. This is valid for AAL2 working in switched mode.
0x32–0x3F		**—**	Reserved. Should be cleared.

32.3.2.3 Determining UEAD_OFFSET (UEAD Mode Only)

The UEAD_OFFSET value described in Table 32-18 is based on the position of the user-defined extended address (UEAD) in the UDC extra header. Table 32-22 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 32-22. UEAD_OFFSETs for Extended Addresses in the UDC Extra Header

32.3.2.4 VCI Filtering (VCIF)

VCI filtering enable bits are shown in Figure 32-28.

	0–2	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 32-28. VCI Filtering Enable Bits

Table 32-23 describes the operation of the VCI filtering enable bits.

Table 32-23. 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.

32.3.2.5 Global Mode Entry (GMODE)

Figure 32-29 shows the layout of the global mode entry (GMODE).

	0–1	2	3–5	6	7	8	9	10	11	12	13	14–15
Field	—	GBL	—	ALB	CTB	REM	MSP	IMA_EN	UEAD	CUAB	EVPT	ALM

Figure 32-29. Global Mode Entry (GMODE)

Table 32-24 describes GMODE fields.

Table 32-24. 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 19.1.8.2, “Serial DMA Mode Register (SDMR).”
3–5	—	Reserved, should be cleared.

Table 32-24. GMODE Field Descriptions (continued)

Bits	Name	Description
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	MSP	MSP Mode. 0 MSP is disabled. 1 MSP is enabled at least on one of the PHYs.
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.
11	UEAD	User-defined cells extended address mode. See 32.2.16.1/32-33. 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 32.2.14/32-25. 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

[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.

32.3.2.6 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 trigger for QUICC Engine block operation. Features are described in Table 32-25.

	0	1	2	3	4	5	6 7	8	9	10	11	12	13 15
Field	ATM_1v1_MT_en	IDLE_ModEn	TSR_Sel	—	—	DYN_ACEn	UID	VPSW_En	NPL	MCAL			—

Figure 32-30. UCC Modes Entry

Table 32-25. UCC_Modes Field Descriptions

Bits	Name	Description
0	**ATM_lvl_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_ModEn	0 No Idle cells are transmitted. This is for normal operation of the ATM where the ATM traffic is on a Utopia bus 1 Enable transmission of idle cells. This is when Serial ATM code is performing the TC layer and Idle cells are required.
2	**TSR_Sel**	QUICC Engine timer prescale select for transmitter scheduling e.g. 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	**Res**	Reserved. Should be cleared
4	—	Reserved, should be cleared.
5	**DYN_ACEn**	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, UCC7 have the UID=0b00, and UCC2, UCC4, UCC6, UCC8 have the UID=0b01. The user should not change this value after issuing Init host command.

Table 32-25. UCC_Modes Field Descriptions (continued)

Bits	Name	Description
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 32.2.14.2, “Mini-CAM Address Look-Up
11		
12		
12–15	—	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 32-26 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.

32.3.2.7 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 32-26 contains the Address Look-Up parameters, It consists of three parameters as described in the table. It is pointed by Table 32-18 sub-page0 configuration table [ADD_COMP_LOOKUP_BASE].

Table 32-26. 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, e.g. 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 32.2.14.1, “Address Compression.” See Figure 32-31 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 32-31 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.

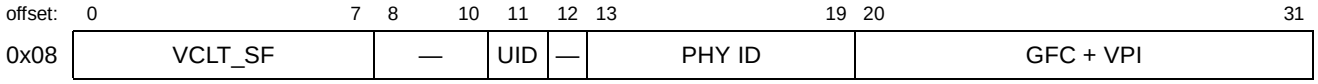


Figure 32-31. VP_LVL_MASK

32.3.2.8 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.

32.3.2.9 Dynamic Address Look-Up Table

This is an identical table to the one described in Table 32-26. This is an alternative table which the user can initialize and use in case of a dynamic address compression change.

32.3.2.10 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 32.2.14.2, "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 32-27. 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 32-33.
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 32-32.
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)
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]

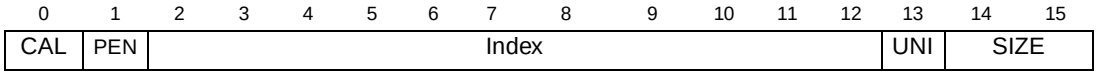


Figure 32-32. PHY Table Entry

Table 32-28. 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

Table 32-28. PHY Table Entry Description (continued)

Bit	Name	Description
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

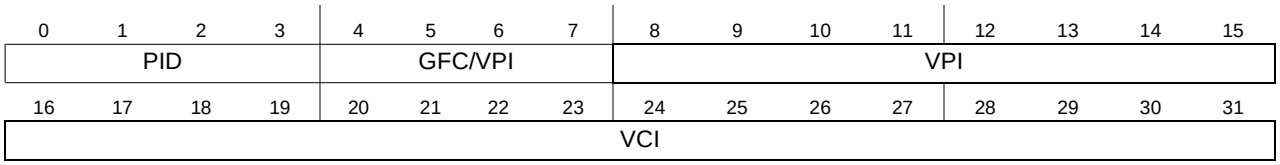


Figure 32-33. Mini-CAM Entry

Table 32-29. 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
16–31	VCI	VCI of the ATM cell

32.3.3 Multi-Threading Structures

When working in Multi-Threading mode, see Section 32.2.3, “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 it’s location in the table. First entry is thread 0, second entry is thread 1 etc. Figure 32-34 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].

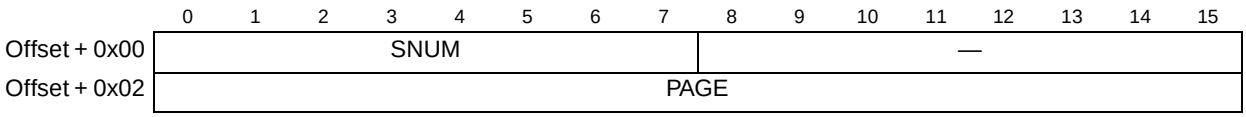


Figure 32-34. ATM Threads Table Entry

Table 32-30. ATM Threads Table Entry Description

Offset	Bits	Name	Description
0x00	0–7	SNUM	Back End SNUM. For available thread numbers see Table 20-15.
	8–15	—	Reserved. Should be cleared.
0x02	—	PAGE	Page pointer.

32.3.4 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 32-35.

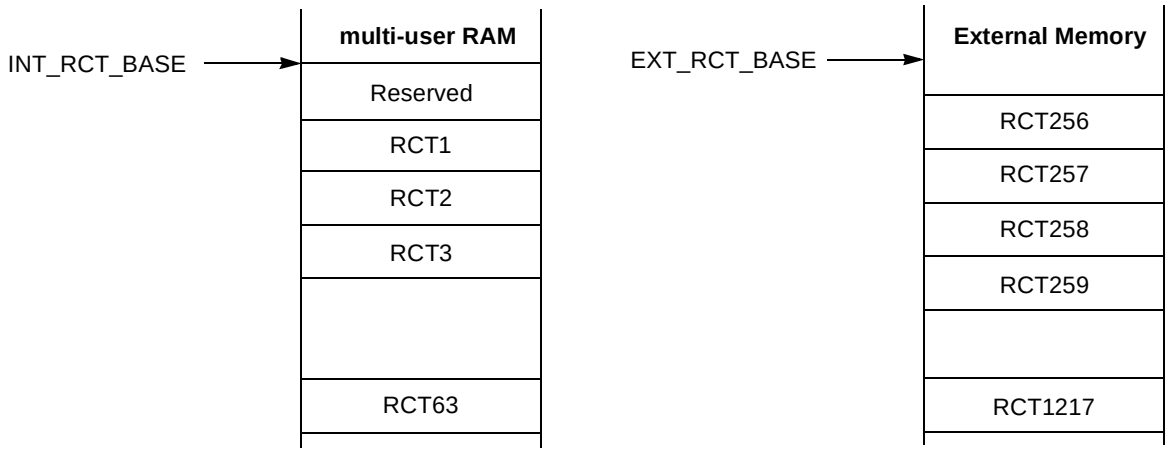


Figure 32-35. Example of a 1024-Entry Receive Connection Table

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:

$$EXT\_RCT\_BASE + (256 \times 32) = EXT\_RCT\_BASE + 8192$$

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 32.3.2, "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.)

32.3.4.1 Receive Connection Table (RCT)

Figure 32-36 shows the format of an RCT entry.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		CETM	DTB	BDB	—	BUFM	SEGF	ENDF	—		INTQ	
Offset + 0x02	—	INF							—				—		AAL	
Offset + 0x04 Offset + 0x06	RX Data Buffer Pointer (RXDBPTR)															
Offset + 0x08	Cell Time Stamp(MSB)															
Offset + 0x0A	Cell Time Stamp(LSB)								Cell Time Stamp(LSB)							
Offset + 0x0C	RBD_Offset															
Offset + 0x0E Offset + 0x10 Offset + 0x12 Offset + 0x14 Offset + 0x16 Offset + 0x18	Protocol Specific • For AAL5,Section 32.3.4.1.1, "AAL5 Protocol-Specific RCT." • For AAL2 RCT Section 41.4.4.1, "AAL2 Receive Connection Tables." • For AAL1, Section 32.3.4.1.2, "AAL1 Protocol-Specific RCT." • For AAL1 CES, AAL1 CES RCT Section 35.9.1.1, "AAL1 CES Protocol-Specific RCT." • For AAL0, Section 32.3.4.1.3, "AAL0 Protocol-Specific RCT."															
Offset + 0x1A	MRBLR															
Offset + 0x1C	—	PMT							RBD_BASE							
Offset + 0x1E	RBD_BASE												—		PM	

Figure 32-36. Receive Connection Table (RCT) Entry

Table 32-31 describes RCT fields.

Table 32-31. 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 19.1.8.2, "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 Figure 19-1 and Section 5.4.3.7, "Debug Configuration."
	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 or AAL1 CES 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 32.3.9.3/32-122. 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 32-31. 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 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol, see Chapter 41, “E2AAL2 Microcode” 101 AAL1_CES—Refer to AAL1, see Chapter 35, “ATM AAL1 Circuit Emulation Service” 110 MSP - Multi ServIce Protocol - switched ATM cells, see Chapter 39, “Enhanced MSP Microcode.” 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 20.3.9, “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		Res	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 $\times$ 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.

32.3.4.1.1 AAL5 Protocol-Specific RCT

Figure 32-37 shows the AAL5 protocol-specific area of an RCT entry.

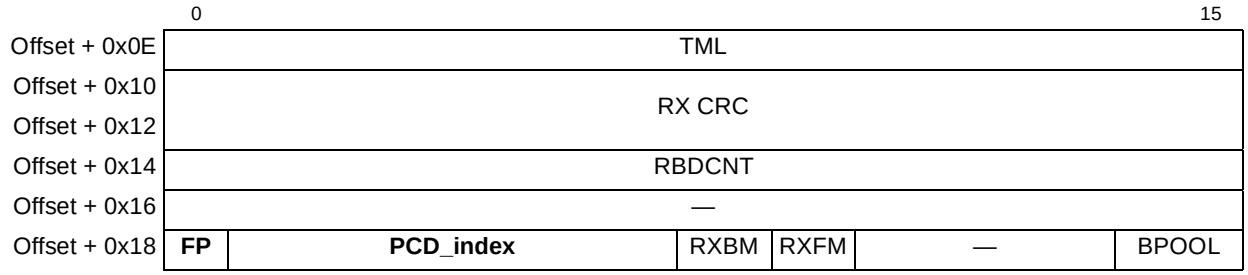


Figure 32-37. AAL5 Protocol-Specific RCT

Table 32-32 describes AAL5 protocol specific RCT fields.

Table 32-32. 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.
	1–7		
	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 32.3.9.2.4/32-120.

32.3.4.1.2 AAL1 Protocol-Specific RCT

Figure 32-38 shows the AAL1 protocol-specific area of an RCT entry.

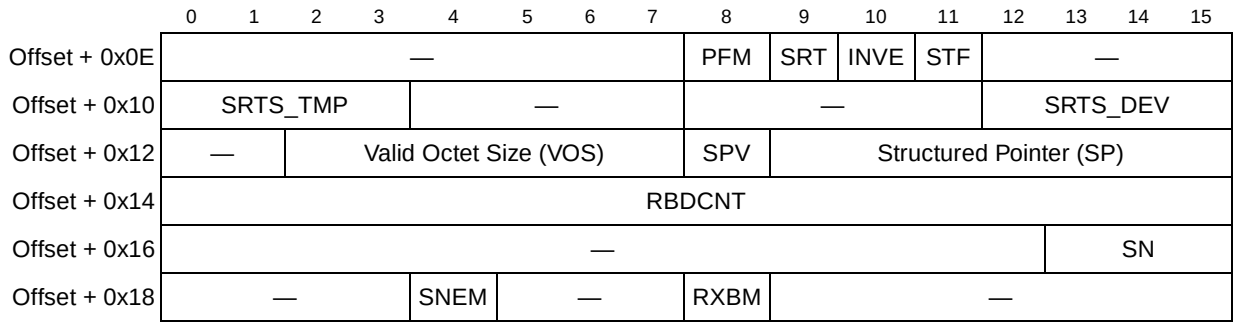


Figure 32-38. AAL1 Protocol-Specific RCT

Table 32-33 describes AAL1 protocol-specific RCT fields.

Table 32-33. AAL1 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–7	—	Reserved, should be cleared.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The QUICC Engine block supports clock recovery using an external SRTS PLL. The QUICC Engine block tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the QUICC Engine block writes a valid SRTS to external logic. (See 32.2.22.4.1/32-48). 0 SRTS mode is not used. 1 SRTS mode is used.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12–15	—	Reserved, should be cleared.
0x10	0–3	SRTS_TMP	Used by the QUICC Engine block to store the received SRTS code. After a cell with SN = 7 is received, the QUICC Engine block writes the SRTS code to the external SRTS device.
	4–11	—	Reserved, should be cleared.
	12–15	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.

Table 32-33. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x12	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only.
	8	SPV	Structured pointer valid. Should be user-initialized user to zero. Structured format only.
	9–15	SP	Structured pointer. Used by the QUICC Engine block to calculate the structured pointer. This field should be initialized by the user to zero. Used in structured format only.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. Initialized with MRBLR whenever the QUICC Engine block opens a new buffer.
0x16	0–12	—	Reserved, should be cleared.
	13–15	SN	Sequence number. Used by the QUICC Engine block to check incoming cell's sequence number.
0x18	0–3	—	Reserved, should be cleared.
	4	SNEM	Sequence number error flag interrupt mask 0 This mode is disabled. 1 When an out-of-sequence error occurs, an RXB interrupt is sent to the interrupt queue even if RCT[RXBM] is cleared. Note that this mode is the buffer error reporting mechanism during automatic data forwarding (ATM-to-TDM bridging) when no buffer processing is required (RCT[RXBM]=0).
	5–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is disabled. (The event is not sent to the interrupt queue.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

32.3.4.1.3 AAL0 Protocol-Specific RCT

Figure 32-39 shows the layout for the AAL0 protocol-specific RCT.

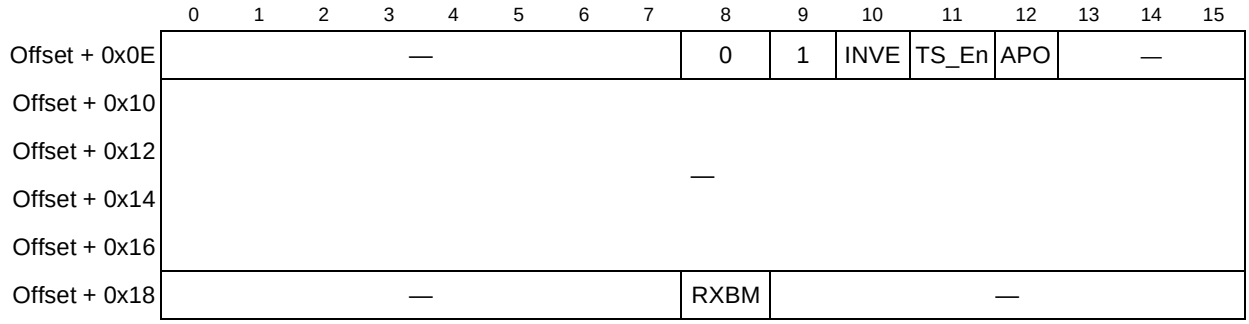


Figure 32-39. AAL0 Protocol-Specific RCT

Table 32-34 describes AAL0 protocol specific RCT fields.

Table 32-34. 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.

32.3.4.1.4 AAL1 CES Protocol-Specific RCT

Refer to Section 35.9.1, “Receive Connection Table (RCT).”

32.3.4.1.5 AAL2 Protocol-Specific RCT

Refer to Section 41.4.4.1, “AAL2 Receive Connection Tables.”

32.3.4.2 Transmit Connection Table (TCT)

Figure 32-40 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 32.3.4.2.1, “AAL5 Protocol-Specific TCT.” • For AAL2, Section 41.3.5.1, “AAL2 Transmit Connection Tables” • For AAL1, Section 32.3.4.2.2, “AAL1 Protocol-Specific TCT.” • For AAL1 CES, Section 35.9.2.1, “AAL1 CES Protocol-Specific TCT” • For AAL0, Section 32.3.4.2.3, “AAL0 Protocol-Specific TCT.”															
Offset + 0x16	APC Linked Channel (APCLC)/GCRA Burst Tolerance (GCRA_BT)															
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 32-40. Transmit Connection Table (TCT) Entry

Table 32-35 describes general TCT fields.

Table 32-35. 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 19.1.8.2, “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 Figure 19-1and Section 5.4.3.7, “Debug Configuration.”

Table 32-35. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	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, AAL1 CES 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 32.3.6.3/32-101, 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 32.3.4.3/32-94.
	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.
		VCON (mode)	Cleared by the host to enable the automatic activation in switch mode - AVCON/AVCF mechanism. When this mode is enabled, the TCT[AVCF] bit should be set. See ATM_AVCON_BASE parameter in Table 32-19 for more description.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 32-35. 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 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol, see Chapter 41, “E2AAL2 Microcode”. 101 AAL1_CES—Refer to AAL1, see Chapter 35, “ATM AAL1 Circuit Emulation Service” 110 MSP - Multi ServIce Protocol - switched ATM cells, see Chapter 39, “Enhanced MSP Microcode” 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 32.2.12.1/32-24 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 32.2.12.1/32-24 for details.
0x10	—	—	Protocol-specific
0x16	—	APCLC/ **GCRA_BT**	APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). When working in GCRA scheduler this field is used to determine the Burst Tolerance allowed for this channel. The user should program the maximum delay allowed between the system time and the last time this channel was active. In case the delay exceeds this value the GCRA scheduler will adjust the channel credit to this value. It determines the maximum burst size for the channel.It is given in TSR Time units. NOTE: For VBR traffic this value should be initialized 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 32-35. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	Res	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 e.g. 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.

32.3.4.2.1 AAL5 Protocol-Specific TCT

Figure 32-41 shows the AAL5 protocol-specific TCT.

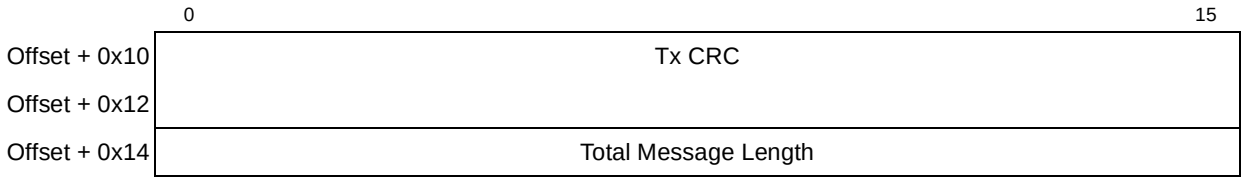


Figure 32-41. AAL5 Protocol-Specific TCT

Table 32-36 describes AAL5 protocol-specific TCT fields.

Table 32-36. 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.

32.3.4.2.2 AAL1 Protocol-Specific TCT

Figure 32-42 shows the AAL1 protocol-specific TCT.

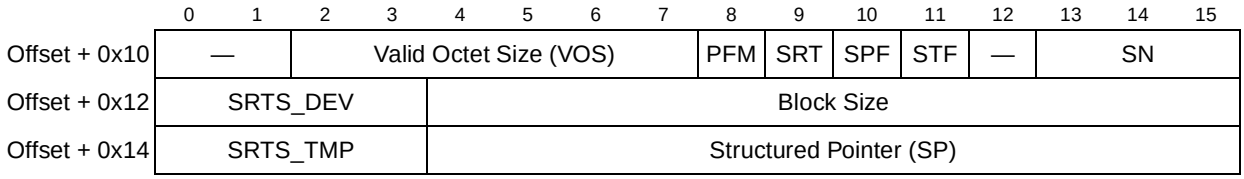


Figure 32-42. AAL1 Protocol-Specific TCT

Table 32-37 describes AAL1 protocol-specific TCT fields.

Table 32-37. AAL1 Protocol-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1-47 are valid; for structured service, values 1-46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The QUICC Engine block supports SRTS generation using external logic. If this mode is enabled, the QUICC Engine block reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The QUICC Engine block reads the new SRTS from external logic every eight cells. (See 32.2.22.4.1/32-48) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPF	Structured pointer flag. Indicates that a structured pointer has been inserted in the current block. The user should initialize this field to zero. Used by the QUICC Engine block only.
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12	—	Reserved, should be cleared.
	13–15	SN	Sequence number field. Used by the QUICC Engine block to generate and maintain the SN of the cell. Should be cleared initially.

Table 32-37. AAL1 Protocol-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28:31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum).
0x14	0–3	SRTS_TMP	Before a cell with SN = 1 is sent, the QUICC Engine block reads the SRTS code from external SRTS logic, writes it to SRTS_TMP, and then inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the QUICC Engine block to calculate the structured pointer. Should be cleared initially. Structured format only.

32.3.4.2.3 AAL0 Protocol-Specific TCT

Figure 32-43 shows the AAL0 protocol-specific TCT.

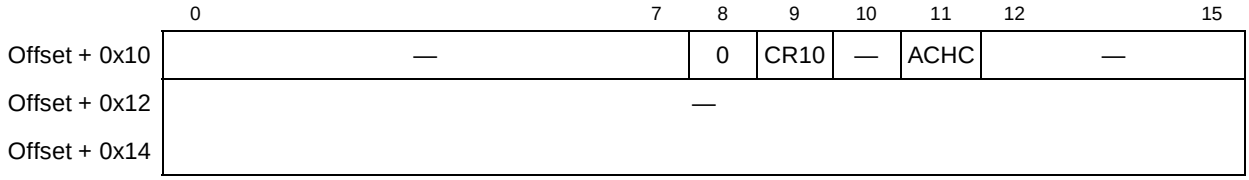


Figure 32-43. AAL0 Protocol-Specific TCT

Table 32-38 describes AAL0 protocol-specific TCT fields.

Table 32-38. 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.

32.3.4.2.4 AAL1 CES Protocol-Specific TCT

Refer to Section 35.9.2, “Transmit Connection Table (TCT).”

32.3.4.2.5 AAL2 Protocol-Specific TCT

Refer to Section 41.3.5.1, “AAL2 Transmit Connection Tables.”

32.3.4.2.6 VBR Protocol-Specific TCTE

Figure 32-44 shows the VBR protocol-specific TCTE.

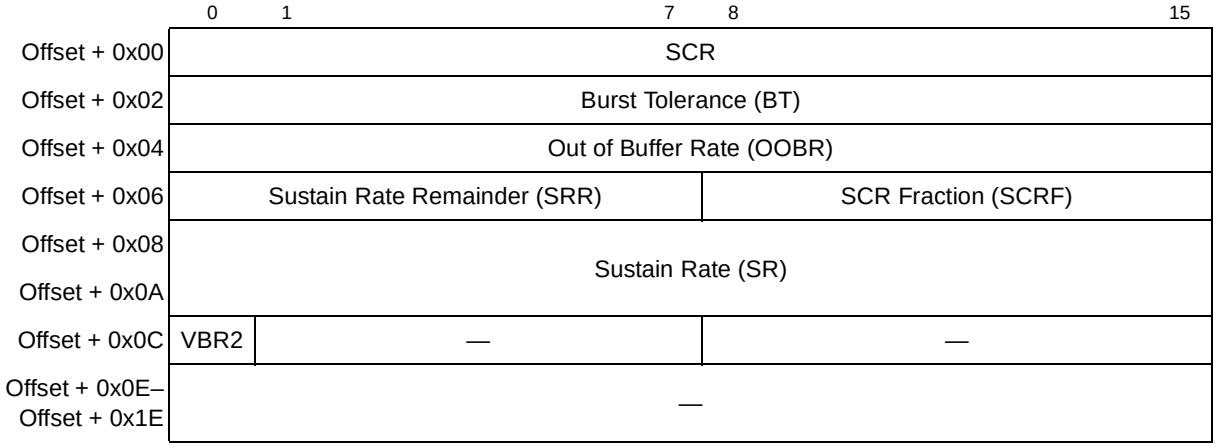


Figure 32-44. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 32-39 describes VBR protocol-specific TCTE fields.

Table 32-39. 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 32.2.12.1, “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 32.2.12.1, “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 32-39. 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	—	res	Reserved, should be cleared.

32.3.4.2.7 UBR+ Protocol-Specific TCTE***

Figure 32-45 shows the UBR+ protocol-specific TCTE.

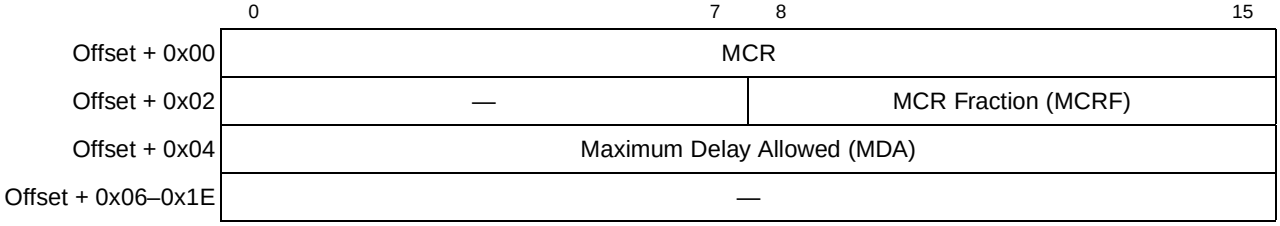


Figure 32-45. UBR+ Protocol-Specific TCTE

Table 32-40 describes UBR+ protocol-specific TCTE fields.

Table 32-40. 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 32.2.12.1, “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.

32.3.4.2.8 Scalable-APC TCTE

Figure 32-46 shows the Scalable-APC TCTE.

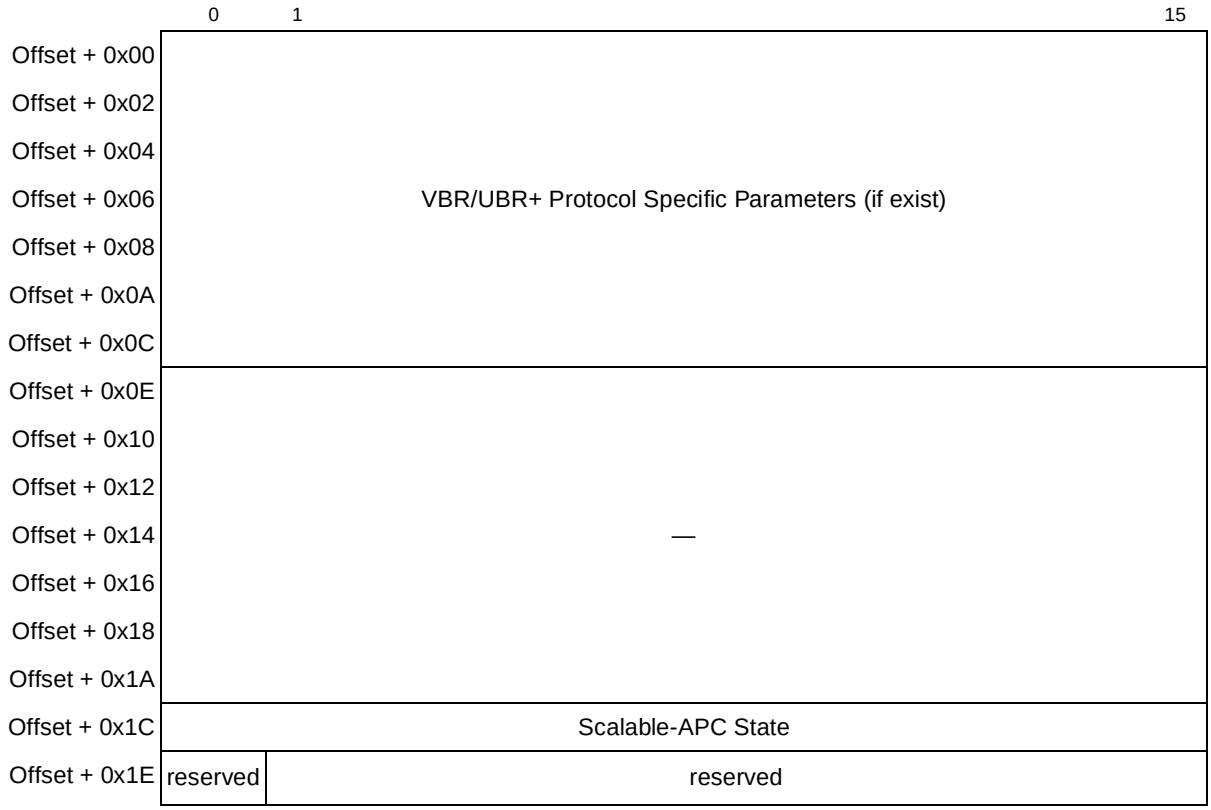


Figure 32-46. Scalable-APC TCTE

Table 32-41 describes Scalable-APC TCTE fields.

Note: This mode doesn't allow working in GBR- Guaranteed Bit Rate mode described in the Section 32.3.4.3, "GBR Programming Model."

Table 32-41. 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.

32.3.4.3 GBR Programming Model

Figure 32-47 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 it's a GBR type. On the highest priority there is no need to set the control slot described in Section 32.3.6.3, "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 32-48.

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.

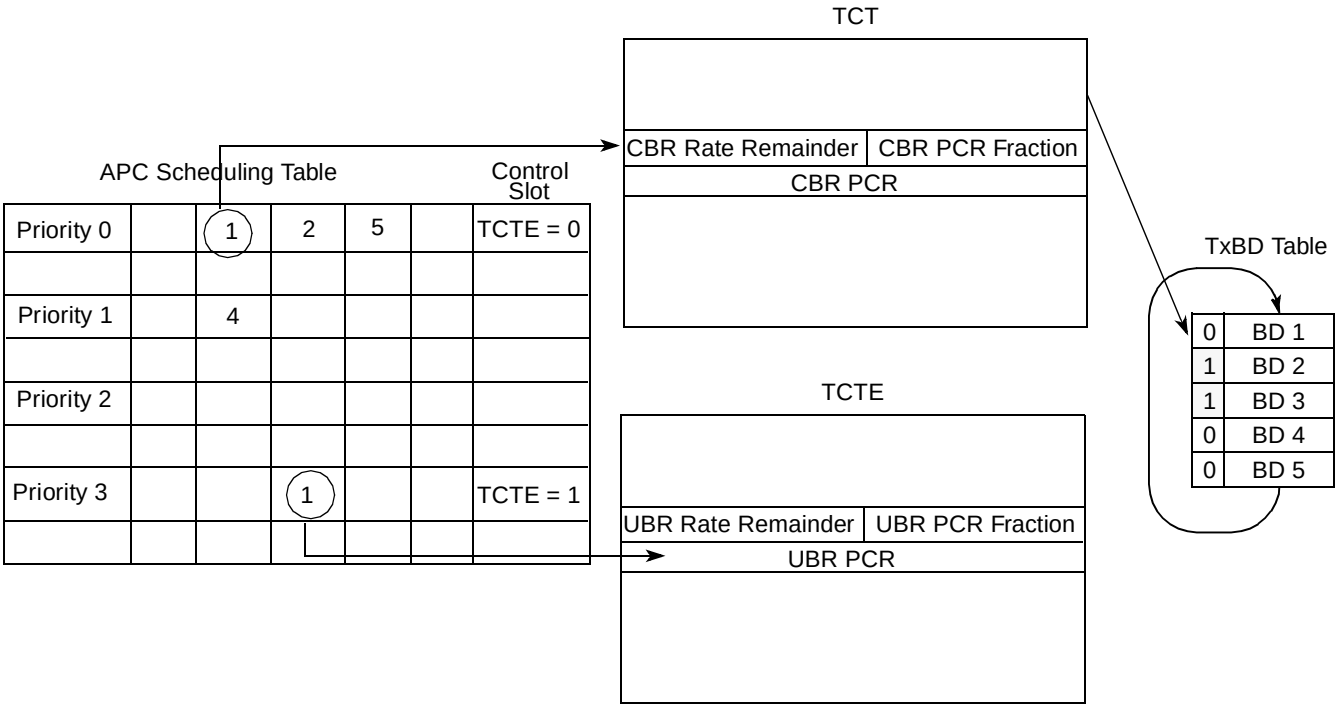


Figure 32-47. GBR Illustration

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 32-70.

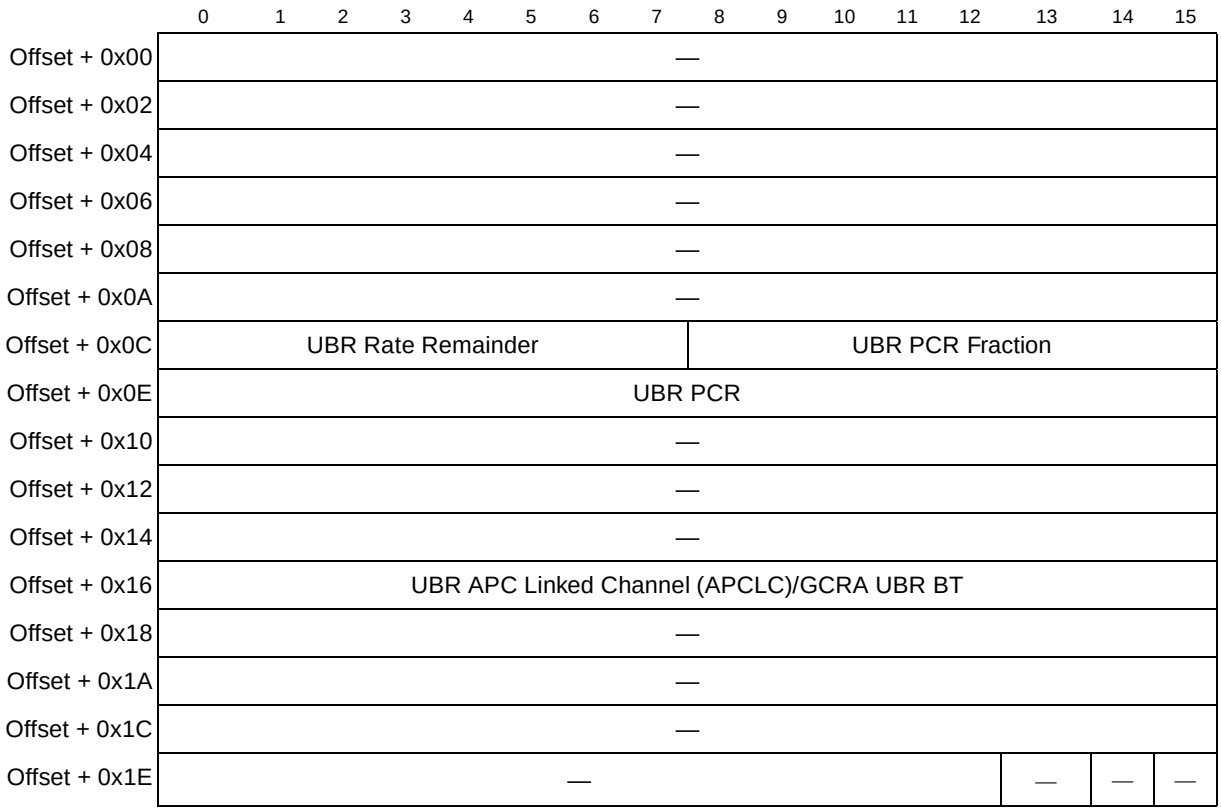


Figure 32-48. GBR Protocol-Specific TCTE

Table 32-42 describes the fields of the GBR protocol-specific TCTE.

Table 32-42. 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 32.2.12.1, “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 32.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x10–0x14	—	—	Reserved, should be cleared.
0x16	—	UBR APCLC/ GCRA UBR BT	APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). For GCRA scheduling this is the BT value for the UBR priority.
0x18	—	—	Reserved, should be cleared.

Table 32-42. GBR Protocol-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
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 32.3.7.1.1, “GBR Operation in GCRA Scheduler Mode.”

32.3.5 OAM Performance Monitoring Tables

The OAM performance monitoring tables include performance monitoring block test parameters, as shown in Figure 32-49. 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 32.2.19, “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.

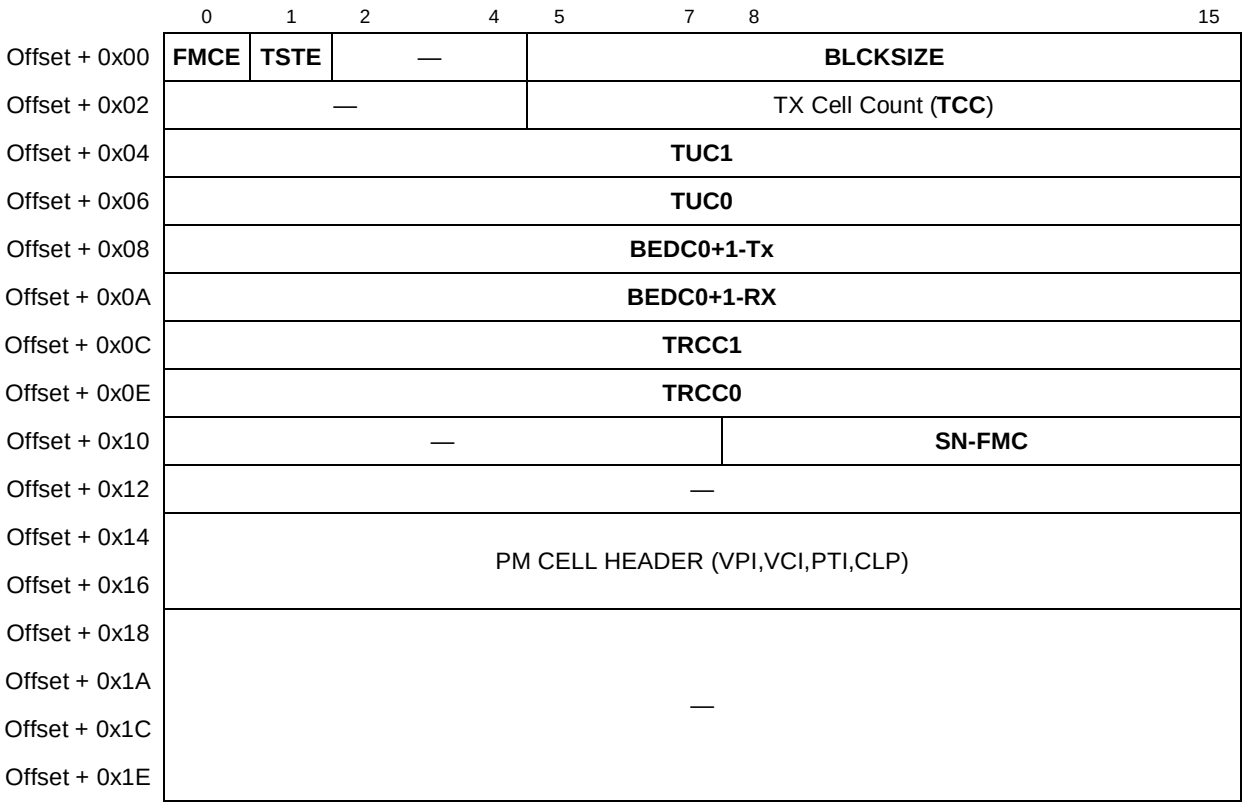


Figure 32-49. OAM Performance Monitoring Table

Table 32-43 describes the fields of the performance monitoring table.

Table 32-43. OAM—Performance Monitoring Table Field Descriptions

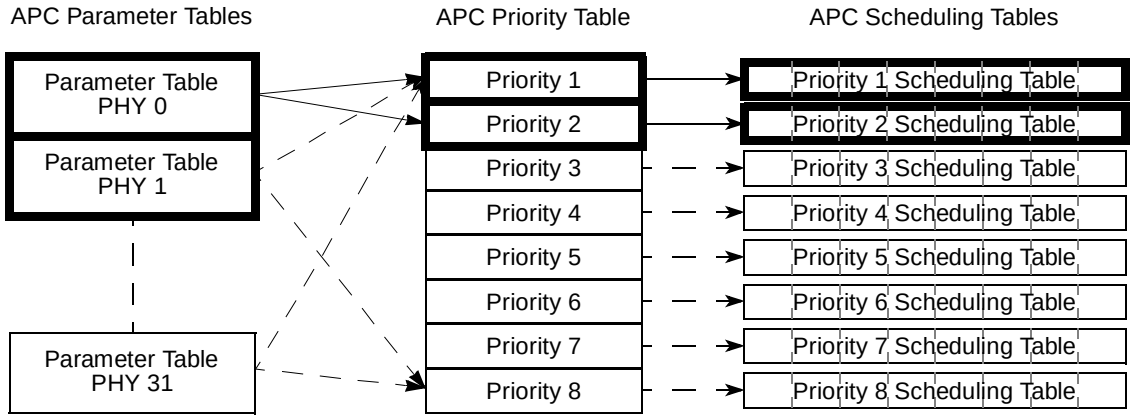
Offset	Bits	Name	Description
0x00	0	**FMCE**	Enables FMC transmission. Initialize to 1.
	1	**TSTE**	FMC time stamp enable 0 The time stamp field of the FMC is coded with all 1's. 1 The value of the time stamp timer is inserted into the time stamp field of the FMC.
	2–4	—	Reserved, should be cleared.
	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 32-43. 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.

32.3.6 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 32-50.



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 32-50. ATM Pace Control Data Structure

32.3.6.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 32-44, 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 32.3.2, "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 situation 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. Note however that in slave mode with multiple PHYs, the parameter table always resides at APCP_BASE regardless of the PHY address.

Table 32-44. 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 32.2.5.2/32-13. 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 32.2.10/32-21, 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 32-45. 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 32.2.8, "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 32-45 Initialized by the user and it represent the fraction of the expected time duration of each APC slot, See description in Section 32.2.8, "APC Flux Compensation."
0x1E	Scheduler_MODE	Hword	Defined in Table 32-45.

[1] Offset values are to APCP_BASE+PHY# × 32. However, in slave mode, the offset is from APCP_BASE regardless of the PHY address.

32.3.6.1.1 Scheduler_MODE

Figure 32-51 shows the layout of the Scheduler_MODE.

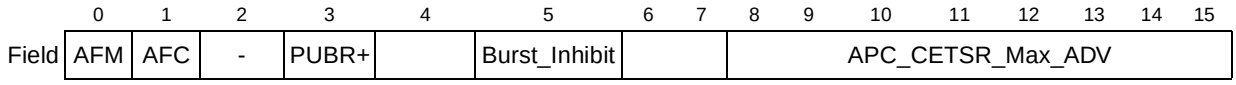


Figure 32-51. Scheduler_MODE

Table 32-45 describes Scheduler_MODE fields.

Table 32-45. Scheduler_MODE Field Descriptions

Bits	Name	Description
0	AFM	Scheduler Mode. if GMODE[MSP]=0) this bit is ignored if GMODE[MSP]=1) Meaning MSP operation is enabled in the system 0- This PHY's scheduling is eight FIFO queues. See "Section 39.10.3.2, Scheduler Parameter Table in WFQ Mode (AFM=0)" on page 53 1- This PHY's scheduling is APC based.
1	AFC	APC Flux Compensation mode. 0 - No time compensation mode. 1 -APC Time Compensation mode is enabled. See 32.2.8/32-19.
2	—	Reserved. Should be cleared.
3	PUBR+	Prioritized UBR+ mode. 0 - Prioritized UBR+ mode is disabled. 1 - Prioritized UBR+ mode is enabled. See 32.2.10/32-21, 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)

32.3.6.2 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 32-46 shows the structure of a priority table entry.

Table 32-46. 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	

32.3.6.3 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 32-52. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

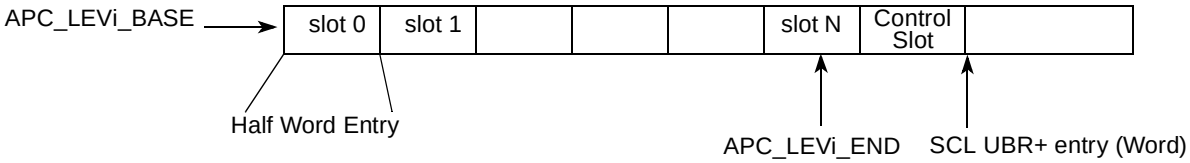


Figure 32-52. The APC Scheduling Table Structure

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 32-53), or the APC overload mechanism for AAL2 described in Section 41.4.3.1.2, “Automatic Bandwidth Allocation” is enabled, 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 32-53

Slot N+1 is used as a control slot, as shown in Figure 32-53.

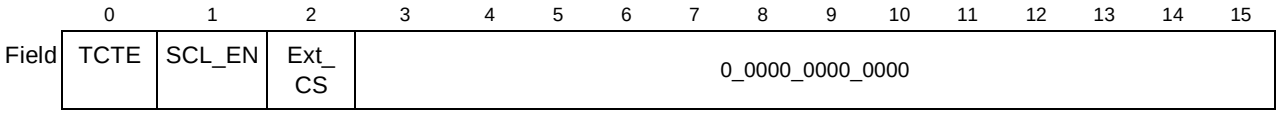


Figure 32-53. Control Slot

Table 32-47 describes control slot fields.

Table 32-47. 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 32.2.7/32-18.
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 or the APC overload mechanism for AAL2 is enabled.
3–15	—	Reserved, should be cleared.

32.3.6.4 UBR+ Priority Decision Table Entry

When the UBR+ priority mode is enabled, the UBR+ priority decision table must be initialized, see Section 32.2.10, “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 32-54, describes the UBR+ priority decision table entry.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	**LSC**	**Res**														
0x02	**THRESHOLD**															

Figure 32-54. UBR+ Priority Decision Table Entry

Table 32-48. UBR+ Priority Decision Table Entry

Offset	Bit	Name	Description
0x00	0	—	Reserved. Should be initialized to 0.
	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).

32.3.7 GCRA Scheduler Structures

32.3.7.1 GCRA Scheduler PHY Parameter Table

The GCRA scheduler PHY Parameter RAM table occupies 32 bytes, see Figure 32-55. 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 32.2.12, “GCRA Scheduler”.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Time Stamp Shadow Register0															
offset + 2																
Offset + 4	Time Stamp Shadow Register1															
offset + 6																
offset + 8	Time Stamp Shadow Register2															
offset + A																
Offset + C	Time Stamp Shadow Register3															
Offset + E																
offset + 10	**GCRA_ SCHED = 1**	**TCTE**	**Expa nd**	**GBR _En**			**Last_ priority**		**Size**							
offset + 12	**FIRST_CC**															
offset + 14	**Res**															
offset + 16	**GCRA_SCHED_Table_ptr**															
offset + 18	**Res**															
offset + 1A	**Res**															
offset + 1C	**Res**															
offset + 1E	**Res**															

Figure 32-55. GCRA Scheduler PHY Parameter Table

Table 32-49. 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 32-49. 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 and Table 32-51 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 32.3.7.1.1, “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. 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	—	**GCRA_SCHEDTable_ptr_**	Word	GCRA Scheduler Table pointer. Initialized to the base address of the GCRA scheduler table. Should be aligned to 32 bytes. The maximum memory allocation for 64 ATM channels in expanded mode is: 32+64*2 = 160 bytes. For a table which has less then 32 channels the size is 16+ Num of channels*2.
0x18	—	—	4*Hword	Reserved. Should be cleared.

[1] Offset from PHY# * 0x20.

Table 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 TCT[PCR] value (e.g. 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.

32.3.7.1.1 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 32-56. GCRA Scheduling Table Structure - Expand =1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Pri0_MASK															
Offset + 2																
Offset + 4																
Offset + 6																
Offset + 8	Pri1_MASK															
Offset + A																
Offset + C																
Offset + E																
Offset + 10	Pri2_MASK															
Offset + 12																
Offset + 14																
Offset + 16																

Figure 32-56. GCRA Scheduling Table Structure - Expand =1 (continued)

Offset + 18	Pri3_MASK
Offset + 1A	
Offset + 1C	
Offset + 1E	
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 32-50. 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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24. 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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24.
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 APC. This can be done only by issuing host command. See32.2.12.2/32-24.

Table 32-50. GCRA Scheduler PHY Scheduling Table Description - Expand=1 (continued)

Offset	Bits	Name	Width	Description
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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24.
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 APC.

Table 32-51. GCRA Scheduling Table structure - Expand=0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	Pri0_MASK															
Offset + 2																
Offset + 4	Pri1_MASK															
Offset + 6																
Offset + 8	Pri2_MASK															
Offset + A																
Offset + C	Pri3_MASK															
Offset + E																
Offset + 10	CH 0 FT															
Offset + 12	CH 1 FT															
Offset + 14	CH 2 FT															
Offset + 16	CH 3 FT															
Offset + 18	CH 4 FT															
Offset + 1A	CH 5 FT															
Offset + 1C	CH 6 FT															
Offset + 1E	CH 7 FT															
Offset + 20 - offset + 47	: : :															
Offset + 48	CH 28FT															
Offset + 4A	CH 29FT															
Offset + 4C	CH 30FT															
Offset + 4E	CH 31FT															

Table 32-52. GCRA Scheduler PHY Scheduling Table description- Expand=0

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 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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24. 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**	DWord	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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24.
0x8	—	**Pri2_MASK**	DWord	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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24.
0x0C	—	**Pri3_MASK**	DWord	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 APC. This can be done only by issuing host command. See 32.2.12.2/32-24.
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 APC.

32.3.8 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 32-57 and it requires data structures as described in the following paragraphs.

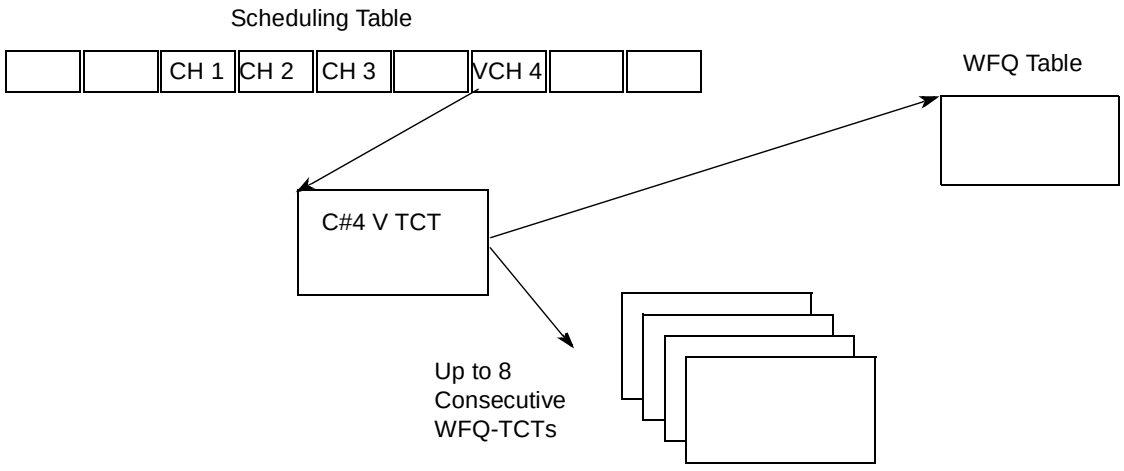


Figure 32-57. Hierarchical Scheduling

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 32-58 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.

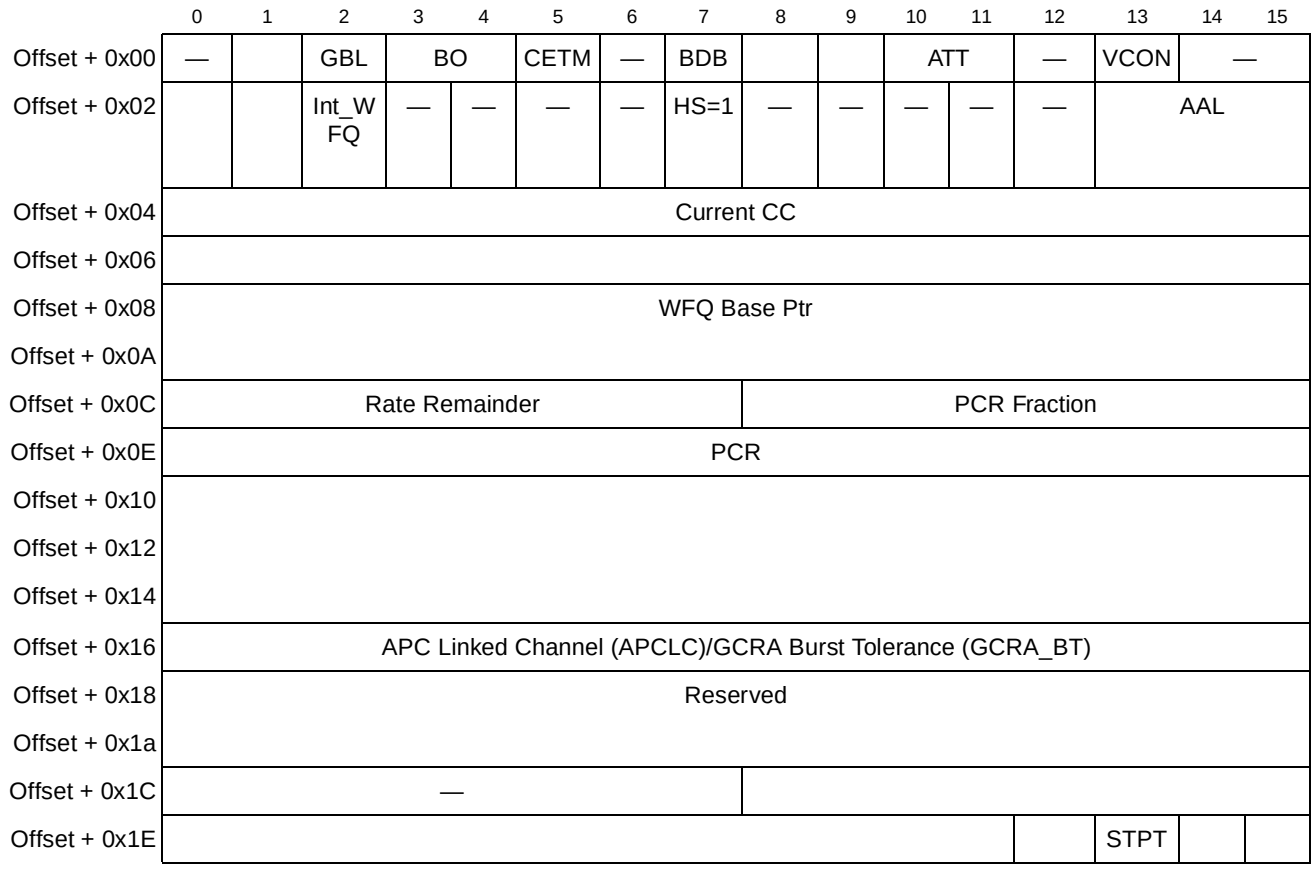


Figure 32-58. Virtual Transmit Connection Table (V-TCT) Entry

Table 32-53 describes the data structure fields.

Table 32-53. 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 19.1.8.2, “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 Figure 19-1.and Section 5.4.3.7, “Debug Configuration.”
	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 32.3.6.3, “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 32.3.4.3, “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.
		VCON (mode)	Cleared by the host to enable the automatic activation in switch mode - AVCON/AVCF mechanism. When this mode is enabled, the TCT[AVCF] bit should be set. See ATM_AVCON_BASE parameter in Table 32-19 for more description.
	14–15	—	Reserved. Should be cleared

Table 32-53. V-TCT Field Descriptions (continued)

Offset	Bits	Name	Description
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
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 32-62.
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 32.2.12.1, “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 32.2.12.1, “GCRA Scheduler Rate Programming” for details.
0x10	—	—	Reserved. Should be cleared.
0x16	—	APCLC/ **GCRA_BT**	APC linked channel. Used by the QUICC Engine block. Initialize to 0 (null pointer). When working in GCRA scheduler this field is used to determine the Burst Tolerance allowed for this channel. The user should program the maximum delay allowed between the system time and the last time this channel was active. In case the delay exceeds this value the GCRA scheduler will adjust the channel credit to this value. It determines the maximum burst size for the channel.It is given in TSR Time units. NOTE: For VBR traffic this value should be initialized to 0xFFFF.
0x18	—	Res	Reserved. Should be cleared

Table 32-53. V-TCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 32-59.

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 32.3.4.2.1, “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 32-59. 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 32-62 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 32-60. 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 32-61. 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 32-59 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.

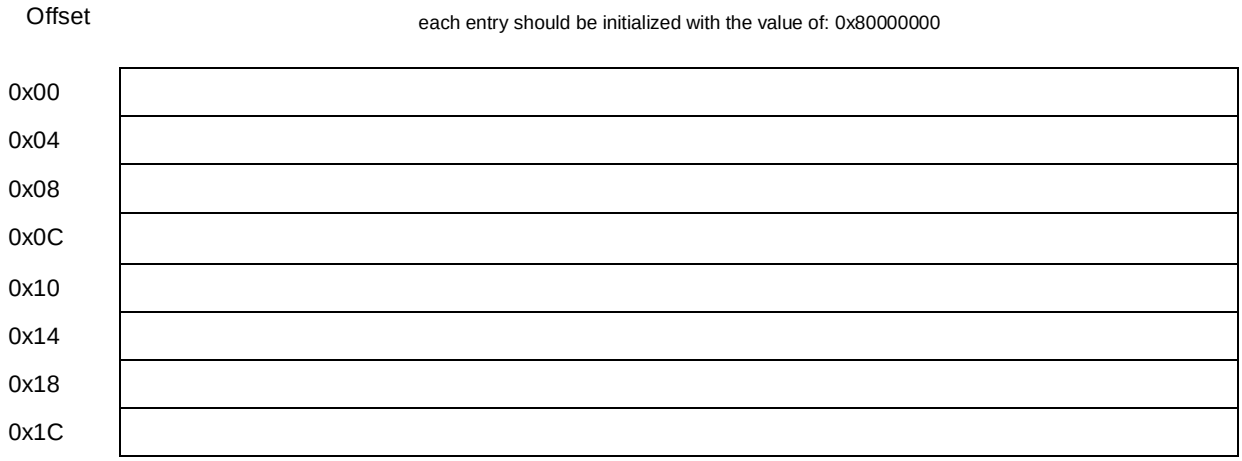


Figure 32-62. WFQ Structure

32.3.8.1 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 32-62
- 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 32.4.1, "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.

32.3.9 ATM Controller Buffer Descriptors (BDs)

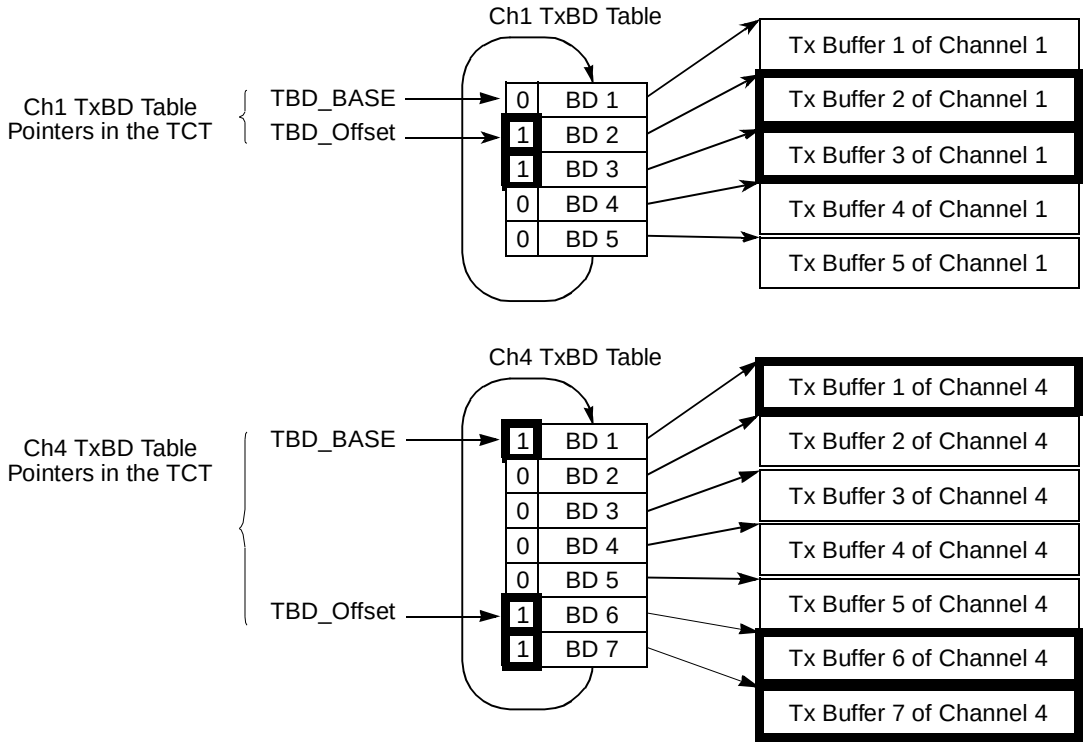
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- 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.

32.3.9.1 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 32-63 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.

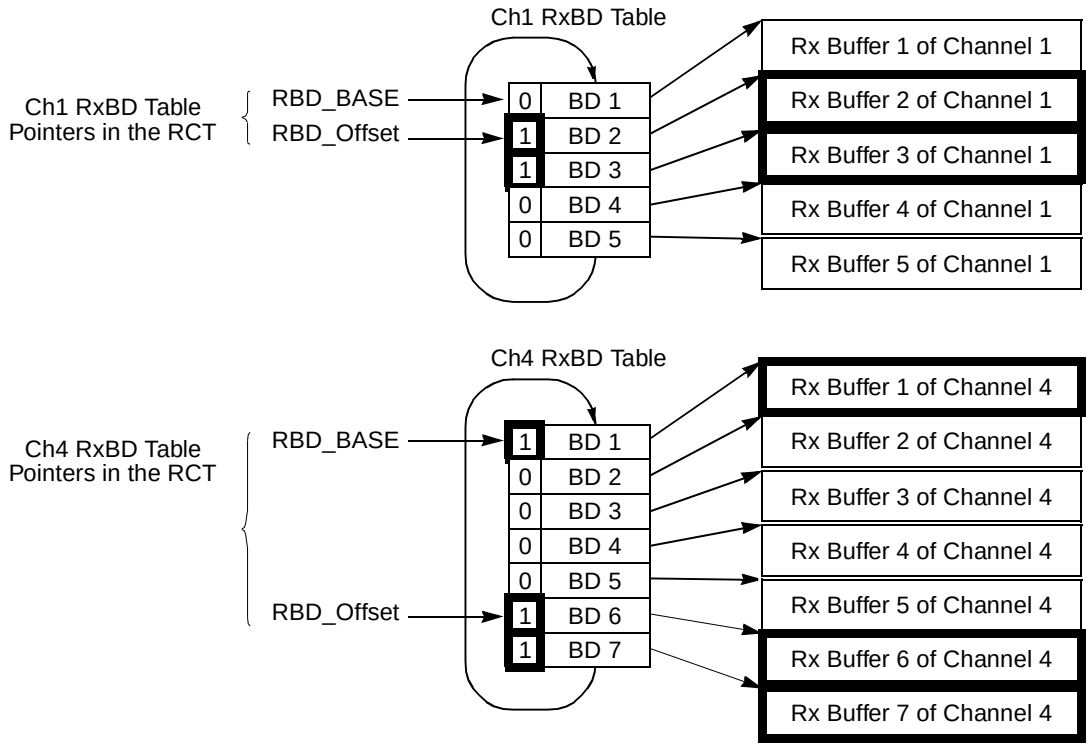
Figure 32-63. Transmit Buffers and BD Table Example

32.3.9.2 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]. AAL1 CES and AAL0 channels must use static buffer allocation.

32.3.9.2.1 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 32-64 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.

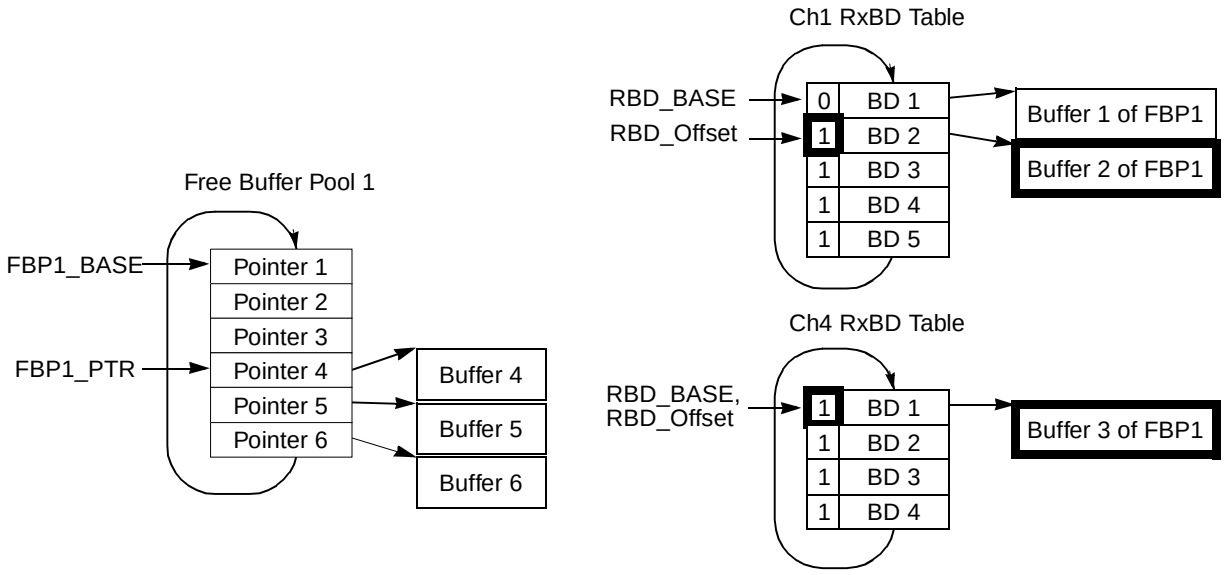
Figure 32-64. Receive Static Buffer Allocation Example

32.3.9.2.2 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 32.3.9.2.3, "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 32-65 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 32-65. Receive Global Buffer Allocation Example

32.3.9.2.3 Free Buffer Pools

As Figure 32-66 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.

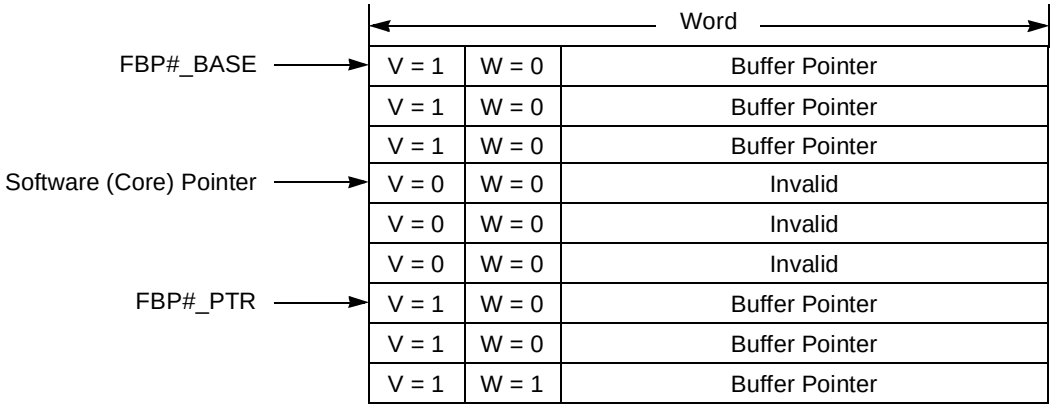


Figure 32-66. Free Buffer Pool Structure

Figure 32-67 describes the structure of a free buffer pool entry.

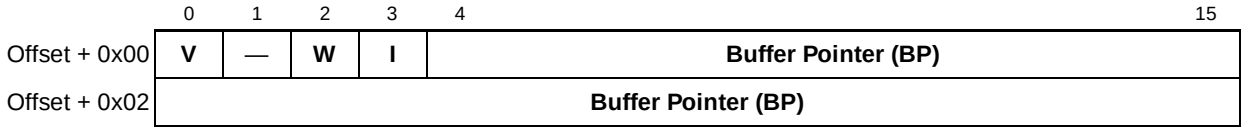


Figure 32-67. Free Buffer Pool Entry

Table 32-54 describes free buffer pool entry fields.

Table 32-54. 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		

32.3.9.2.4 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 s/w behavior. In Non Multi-threading mode the operation and programming model is identical to this of the PQII family and is depicted in Table 32-55.

Table 32-55. 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 32-55. 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 reenable 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 32-56. 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 32-56. 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		**Res**	**HW**	Reserved- should be cleared

1 Offset from FBT_BASE+RCT[BPOOL] × 16

The maximum size of the FBP is 16K entries for Multi-thread mode.

32.3.9.3 ATM Controller Buffers

Table 32-57 describes properties of the ATM receive and transmit buffers for better bus utilization.

Table 32-57. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	Alignment	Size	Alignment
AAL5	Multiple of 48 octets (except last buffer in frame)	Double word aligned	Any	No requirement
AAL1	At least 47 octets	No requirement	At least 47 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

32.3.9.4 AAL5 RxBD

Figure 32-68 shows the AAL5 RxBD fields.

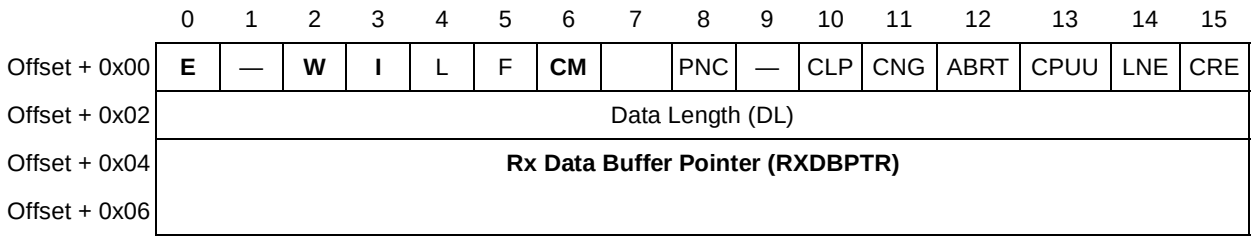


Figure 32-68. AAL5 RxBD

Table 32-58 describes AAL5 RxBD fields.

Table 32-58. 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		
	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 32-58. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
	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.

32.3.9.5 AAL1 RxBD

Figure 32-69 shows the AAL1 RxBD.

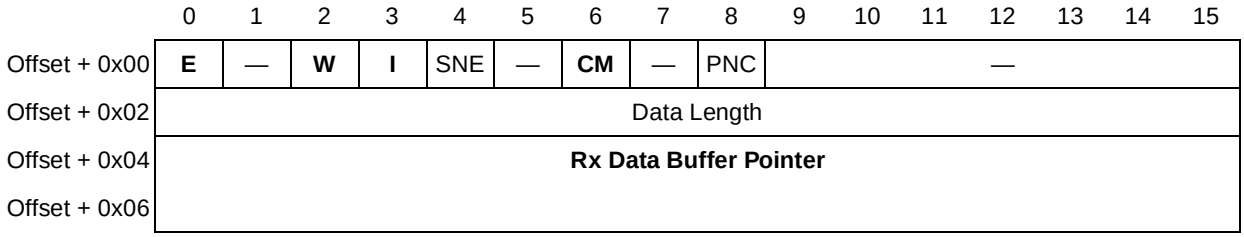


Figure 32-69. AAL1 RxBD

Table 32-59 describes AAL1 RxBD fields.

Table 32-59. AAL1 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 read or write any fields of this RxBD. The QUICC Engine block cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the 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 is 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 overall space is constrained to 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. UCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4	SNE	Sequence number error. SNE is set when a sequence number error is detected in the current AAL1 CES buffer.
	5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation. 1 The empty bit (RxBD[E]) is not cleared by the QUICC Engine block 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 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.
	9–15	—	Reserved, should be cleared.
0x02	—	DL	Data length. The number of octets the QUICC Engine block writes into the buffer once its BD is closed.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. For better bus performance this pointer should be burst-aligned.

32.3.9.6 AAL0 RxBD

Figure 32-70 shows the AAL0 RxBD.

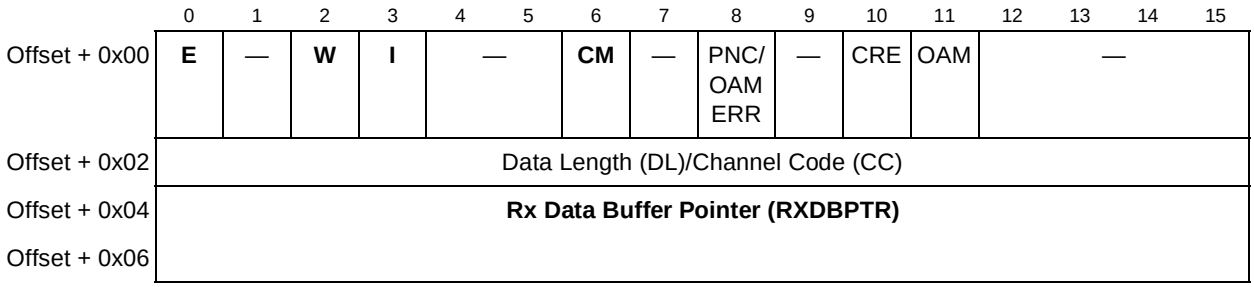


Figure 32-70. AAL0 RxBD

Table 32-60 describes AAL0 RxBD fields.

Table 32-60. 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.

Table 32-60. AAL0 RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
	8	PNC/ OAMERR	UPC Non Conformance/OAMERR. If RxBD[OAM] is set (and MSP mode is enabled), this field functions as OAMERR; otherwise it is PNC. 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. OAMERR - When MSP is enabled, and an erroneous OAM cell received by the switch, them this bit is set by the QUICC Engine block. See Chapter 39, "Enhanced MSP Microcode."
	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.

32.3.9.7 AAL1 CES RxBD

Refer to Section 35.12.1, "AAL1 CES RxBD."

32.3.9.8 AAL2 RxBD

Refer to Section 41.4.4.6, "CPS Receive Buffer Descriptor (RxBD)."

32.3.9.9 AAL5, AAL1 CES User-Defined Cell—RxBD Extension

In user-defined cell mode, the AAL5and AAL1 CES RxBDs are extended to 32 bytes; see Figure 32-71.

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 32-71. User-Defined Cell—RxBD Extension

32.3.9.10 AAL5 TxBDs

Figure 32-72 shows the AAL5 TxBD.

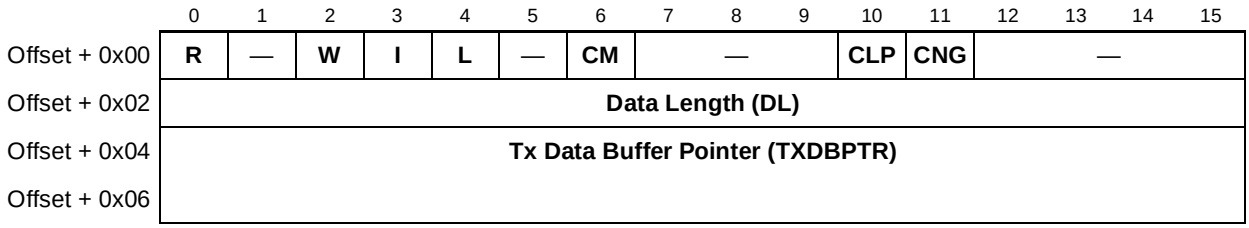


Figure 32-72. AAL5 TxBD

Table 32-61 describes AAL5 TxBD fields.

Table 32-61. 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.

32.3.9.11 AAL1 TxBDs

Figure 32-73 shows the AAL1 TxBD.

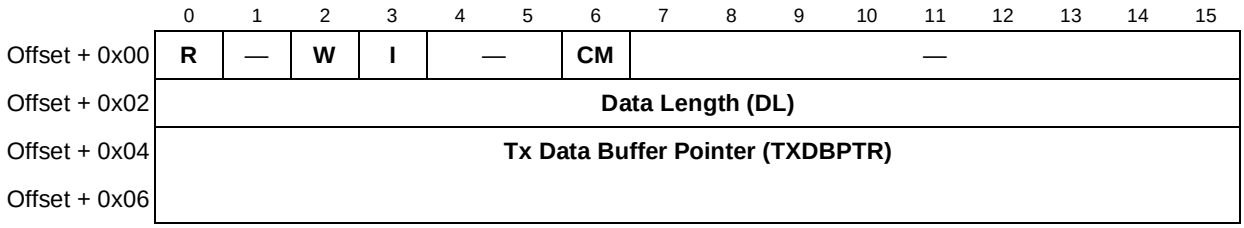


Figure 32-73. AAL1 TxBD

Table 32-62 describes AAL1 TxBD fields.

Table 32-62. AAL1 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The QUICC Engine block clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the 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.
	4–5	—	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.
	7–11	—	Reserved, should be cleared.
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the 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. The buffer may reside in either internal or external memory. This value is not modified by the QUICC Engine block.

32.3.9.12 AAL0 TxBDs

Figure 32-74 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.

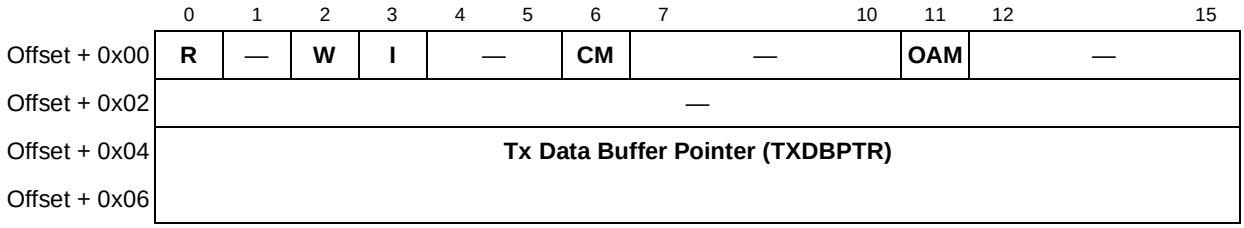


Figure 32-74. AAL0 TxBDs

Table 32-63 describes AAL0 TxBD fields.

Table 32-63. 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 32-63. 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.

32.3.9.13 AAL1 CES TxBDs

Refer to Section 35.12.2, "AAL1 CES TxBDs."

32.3.9.14 AAL2 TxBDs

Refer to Section 41.3.5.6, "SSSAR Transmit Buffer Descriptor."

32.3.9.15 AAL5, AAL1 User-Defined Cell—TxBD Extension

In user-defined cell mode, the AAL5 and AAL1 TxBDs are extended to 32 bytes, see Figure 32-75. 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.

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 32-75. User-Defined Cell—TxBD Extension

32.3.10 UPC Structures

32.3.10.1 UPC Table

The UPC Table structure is depicted in Figure 32-76. The field "offset" refers to the address generated in the following way (see PID description in Table 32-5 and Table 32-2):

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.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset+0x00										**NCCLPIE**	**CLPIE**	**FCIE/ CDCIE**	**CLPDM**	**IND**	**UPCM**	
Offset+0x02	—															
Offset+0x04	CLP0															
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 32-76. UPC Table

[1] Bolded parameters are user initialized.

Table 32-64 describes UPC table fields.

Table 32-64. 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 32.2.20.2.1/32-43. 10 Frame awareness mode. See 32.2.20.2.2/32-43 (applied only for AAL5). 11 GFR mode. See 32.2.20.2.3/32-43. 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 32-64. 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.

Note: When working with AAL2 traffic DM1 and DM2 should always be set. Tagging mode is not available.

32.3.11 AAL1 Sequence Number (SN) Protection Table

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in multi-user RAM and is used for AAL1 only. The table should be initialized according to Figure 32-77.

	0 – 15
Offset + 0x00	0x0000
Offset + 0x02	0x0007
Offset + 0x04	0x000D
Offset + 0x06	0x000A
Offset + 0x08	0x000E
Offset + 0x0A	0x0009
Offset + 0x0C	0x0003
Offset + 0x0E	0x0004
Offset + 0x10	0x000B
Offset + 0x12	0x000C
Offset + 0x14	0x0006
Offset + 0x16	0x0001
Offset + 0x18	0x0005
Offset + 0x1A	0x0002
Offset + 0x1C	0x0008
Offset + 0x1E	0x000F

Figure 32-77. AAL1 Sequence Number (SN) Protection Table

32.3.12 UNI Statistics Table

The UNI statistics table, shown in Table 32-65, 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 32-65. 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 (i.e. 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

32.3.13 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 32.3.13.3, "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; see Section , "". 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.

32.3.13.1 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 32-78 shows UCCE and UCCM bits.

Offset: Access: User read/write

	0	1	2	3	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 32-78. UCC Event Register (UCCE)/Mask Register (UCCM)

Table 32-66 describes UCCE fields.

Table 32-66. 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 32.3.13, "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 32.3.13.2, "Interrupt Queues."
17–19	—	Reserved, should be cleared.
21–26	—	Reserved, should be cleared.
28–30	—	Reserved, should be cleared.

32.3.13.2 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in a table. See Table 32-68 and Table 32-69

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 32-79.

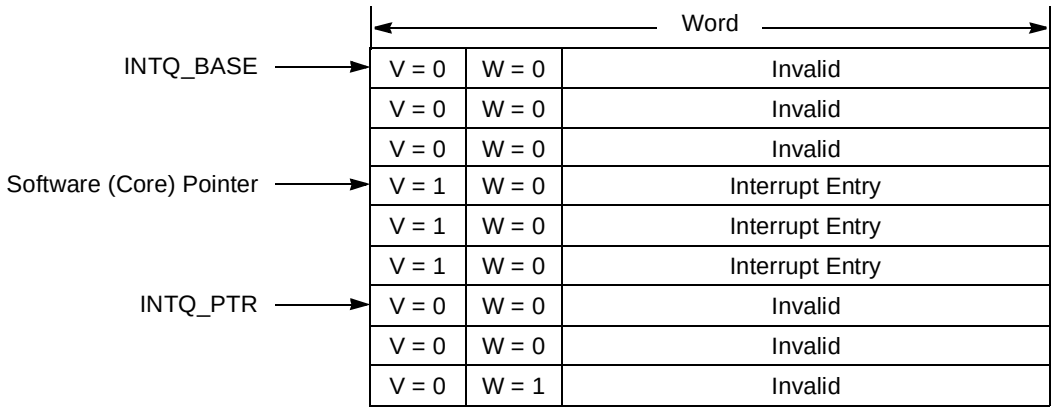


Figure 32-79. Interrupt Queue Structure

32.3.13.3 ATM Termination Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 32-80 shows an entry for Non UPC- Policer events

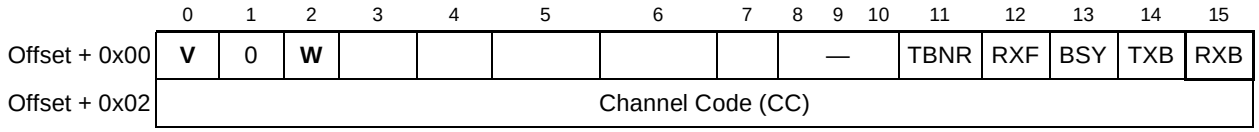


Figure 32-80. Interrupt Queue Entry—Non UPC

Figure 32-81 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.

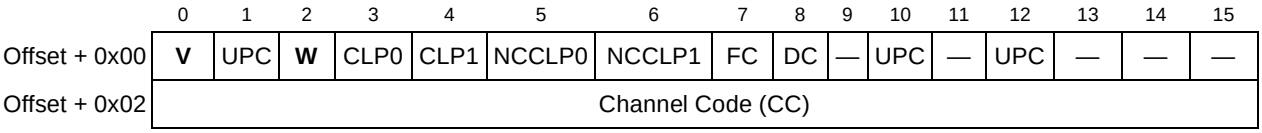


Figure 32-81. Interrupt Queue Entry—UPC Event

Important: For AAL2 there are different interrupt queue entries. See “Section 41.8, AAL2 Exceptions” on page 60 for details.

32.3.13.4 Interrupt Queue Parameter Table

Table 32-67 describes interrupt queue entry fields.

Table 32-67. 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.
	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.

Table 32-67. Interrupt Queue Entry Field Description (continued)

Offset	Bits	Name	Description
0x00	5	NCCLP 0	NCCLP0 counter overflow.Valid only if UPC bit is set. 0 No NCCLP0 counter overflow. 1 NCCLP0 counter overflow.
	6	NCCLP 1	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 32-68 and Table 32-69 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] $\times$ 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 32-68. 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.

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 32-69. 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	**Res**	Half Word	Reserved. Should be cleared
0x0E	**INTQ_Size**	Half Word	Size in bytes of the interrupt queue entries table.

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.

32.4 ATM Controller—Application Information

32.4.1 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:

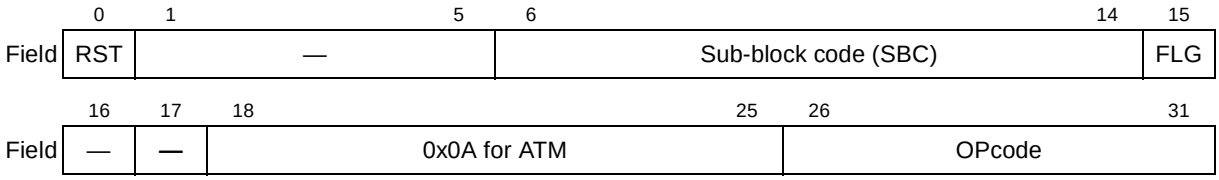


Figure 32-82. CE Command Register (CECR)

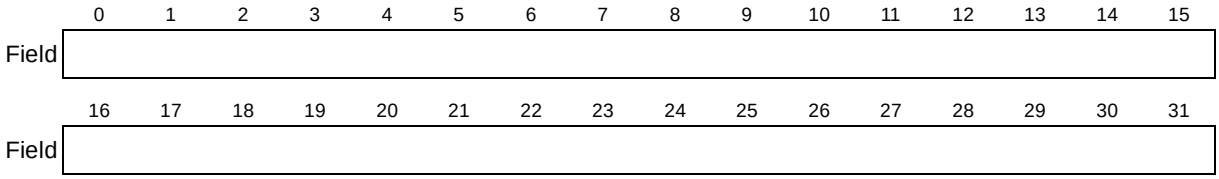


Figure 32-83. CE Command Data Register (CECDR)

The host commands for the ATM are the following

32.4.1.1 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 32-19 The information is described in Figure 32-84.

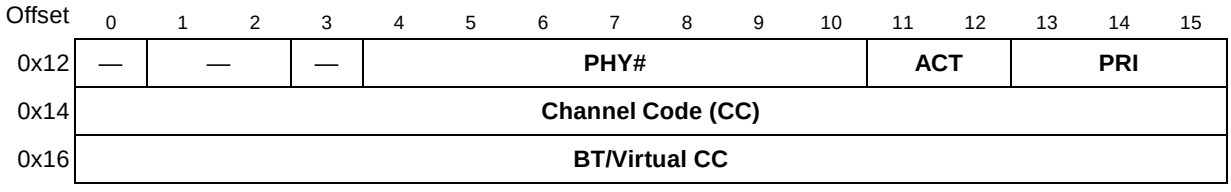


Figure 32-84. COMM_INFO Field

NOTE
Backward Compatibility

The PHY# and the CTB bits locations were changed.

Table 32-70 describes COMM_INFO fields.

Table 32-70. 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#	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 priority level. 000 Highest priority (APC_LEVEL1) 111 Lowest priority (APC_LEVEL8). For GCRA scheduler only 4 priority levels are available. 0x00–0x03
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, e.g. 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.

32.4.1.2 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.

32.4.1.3 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 32-20 and Table 32-30 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.

32.4.2 Configuring the ATM Controller for Maximum QUICC Engine Performance

The following sections recommend ATM controller configurations to maximize QUICC Engine performance.

32.4.2.1 Configuration of Internal Rates timers

Use the transmit internal rate mode and configure the internal rate clock to the maximum bit rate required. The PHY then automatically fills the unused bandwidth with idle cells, not the ATM controller.

For example, suppose a system uses a 155.52-Mbps OC-3 device as PHY0, but the maximum required data rate is only 100 Mbps. In transmit internal rate mode, the user can configure the internal rate mechanism to clock the ATM transmitter at a cell rate of 100 Mbps. If the system clock is 133 MHz, program a BRG to divide the system clock by 563 to generate a transmit cell request every 563 QUICC Engine clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{100\text{Mbps}} = 563$$

Set FTIRR*x*_PHY0[TRM] to enable the transmit internal rate mode and clear FTIRR*x*_PHY0[Initial Value] since there is no need to further divide the BRG.

32.4.2.2 APC Configuration

32.4.2.2.1 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 32.2.5.3.1, “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.

32.4.2.2.2 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 32.2.11, “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.

32.4.2.2.3 APC Flux Compensation

For variable bit rate PHY (e.g. 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 32.2.8, "APC Flux Compensation".

The penalty for using the APC Flux Compensation mechanism is in terms of QUICC Engine performance.

32.4.2.2.4 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.

32.4.2.2.5 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.

32.4.2.3 GCRA Scheduler mode

As described in Section 32.2.12, "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.

32.4.2.4 ATM Multi-thread Mode

As described in Section 32.2.3, "ATM Multi-Threading", in order to support higher ATM traffic rates, as overall bandwidth of OC-12, 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.

32.4.2.5 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 32.3.9.2, "Receive Buffer Operation.") When allowed by the application, use static buffer allocation to increase QUICC Engine performance.

Chapter 33
UTOPIA POS Bus Controller (UPC)

33.1 Overview

The UPC (UTOPIA/POS-PHY L2 bus Controller) is the UTOPIA/POS-PHY MAC peripheral of the QUICC Engine block. The QUICC Engine block supports UTOPIA/POS-PHY level 2 for both master and slave modes.

The QUICC Engine block has two UPCs, referred to as UPC1 and UPC2. In this document, the term "UPC" applies to both. In some diagrams, a reference for a specific UPC is made for clarity, but unless stated otherwise, the UPCs have the same functionality[1].

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 block which are integrated within the UCC, the UPC can route the data path to 1-4 UCCs[2] (see Figure 33-1, routing example in Figure 33-3). This routing is required for certain bus configurations in order to achive the required bus performance.

As master, the UPC controls one UL2/PL2 bus with up to 124/128 PHYs. The UPC implements Appendix 1 to the UTOPIA L2 standard with 4 Clav/PxPA and Enb signal pairs. The group of PHYs controlled by a common Clav/Enb signals pair are called a device within this chapter. Each device could be either a multi PHY or a single PHY. Each device can be routed to any UCC in multi PHY (MPHY) configuration. Only a single device can be routed to any UCC in single PHY (SPHY) configuration.

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. This mechanism enables the allocation of 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 derived from a baud rate generator or from an external clock source. The Internal Rate works differently for SPHY and for 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 (the polling status is qualified by the internal rate, and the FIFO only contains cells for PHYs which 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. For POS, the Internal rate has another option, to control the packets per second rate.

1. Except that the two UPCs might have different I/O options, please check with the parallel I/O specification.
2. UPC1 can be routed only to UCC1, UCC3, UCC5, UCC7., UPC2 can be routed only to UCC2, UCC4, UCC6, UCC8.

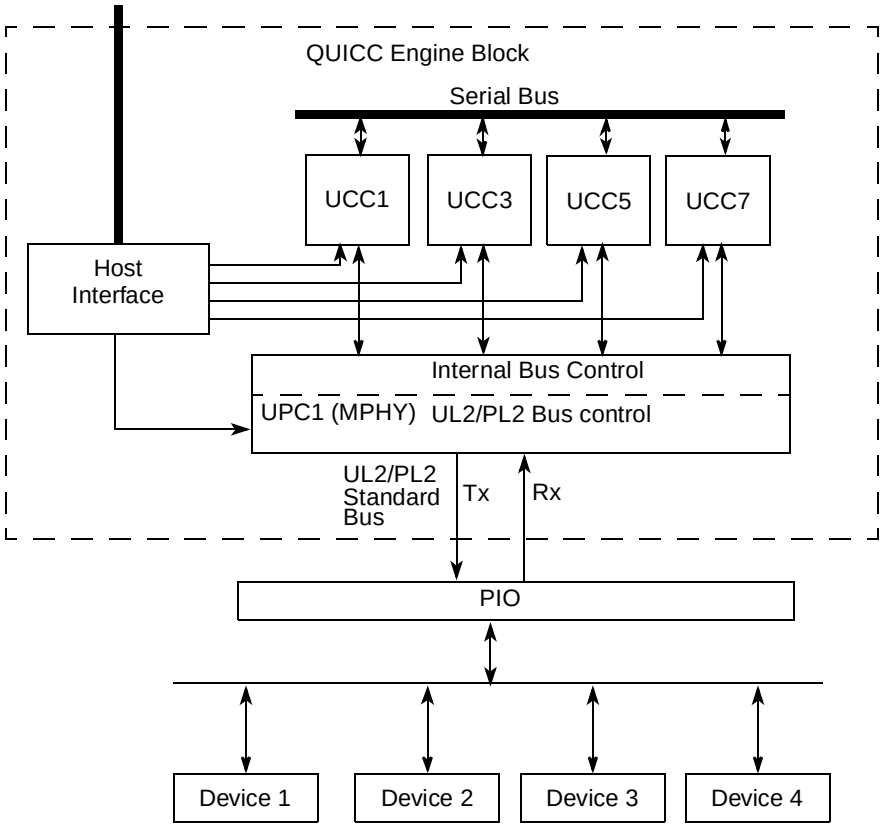


Figure 33-1. QUICC Engine Block with UPC1 (4 devices, up to 124 ports)

33.2 Features

33.2.1 UPC Features

- Connection to 4 devices through one UL2/PL2[1] Standard bus I/F (compliant with appendix 1 of UTOPIA L2 spec).
- UL2 protocol:
  - Up to 124 addresses for 4 physical devices in normal address mode (5-bit address).
  - Up to 128 addresses for 8 physical devices in 6-bit addressing MPHY mode.
  - Connection to 4 UCCs, each with deep, programmable size FIFOs.
  - Internal Loop-back mode
- PL2 protocol:
  - Up to 31 addresses for 1 physical devices in normal address mode (5-bit address).
  - Up to 32 addresses for 2 physical devices in 6-bit addressing MPHY mode.
  - No internal loop-back mode.
- Routing of each device to each UCC in MPHY

1. POS supports only a single device.

- Routing of a single device to a UCC as SPHY
- Slave can tri-state its outputs when not selected (Slave in MPHY system)

NOTE

To maximize performance, the UTOPIA/POS-PHY bus bandwidth should be at least 1.1 times the aggregated throughput on the bus.

33.2.2 UL2 Features

- Conforms to ATF UL2 standard version 1.0, June 1995
- 8/16 bit UL2 bus clocked at 50 MHz with 4 devices (50 pF)
- Cell level handshake support
- Up to 4 devices on the UL2 bus, each with separated Clav/Enb (Per Rx/Tx). Each could be set to either 8 or 16 bit data port width.
- SPHY: Up to 4 devices can be configured as SPHY. Clav is assumed as direct status or as always valid.
- MPHY: Each device can be configured as MPHY, with single Clav polling method per device (there is a separate Clav/Enb control pair for each device).
- Default address mode (5-bit address) for 124 PHYs (4 devices, each with up to 31 PHYs).
- Extended address mode (6-bit address) for 128 PHYs (8 Devices, each with up to 16 PHYs). The devices in extended mode are standard compliant (5-bit address).
- Internal Rate:
  - 4 internal rates for each device (16 total)
  - Programmable max credit value for each device (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 within the same device. Arbitration is in round robin order among the devices routed to the same UCC.
  - Transmit arbitration is handled in round robin among different UCCs.
  - 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.
  - Optional back to back transmission per UCC. This feature is useful for SPHY, for which back to back cell transmission is desired. It is recommend to use this option for PHYs that are allocated more then 50% of the bus BW.
- 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 (common to all devices).
  - Automatic data parity generation.
- Rx selection (two priority levels decision):
  - Arbitration is in round robin priority among devices with optional back to back reception per UCC to prioritize a UCC (in SPHY mode this is a PHY level back to back).
  - 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.
  - Parallel polling over 4 devices in single-Clav method.
  - Maskable polling per port (Rx, Tx or both).
- Slave mode
  - Slave is a single device, 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).

33.2.3 PL2 Features

33.2.4 PL2 Features

- Conformal to Saturn Group PL2 specification, November 1998.
- 16 bit PL2 bus clocked at 50 MHz with 1 device (50 pF).
- Packet level handshake support.
- SPHY: device can be configured as SPHY. PxPA is assumed as always valid or as a direct status polling indication (DxPA).
- Packet interleaving among multiple PHYs in transmit and receive (switching between ports during packets transfer).
- As master, user should configure the transmit segment size to be less or equal to the PHY's PTPA threshold.

- Internal rate mechanism to set the packets per second (PPS) rate.
- Generate TxErr in case of Tx underrun.
- Reports Overrun, RxErr, Parity error, Protocol errors.
- MPHY: Each device can be configured as MPHY, with single PxPA polling status indication per device.
- Default address mode (5-bit address) for 31 PHYs (1 device).
- Extended address mode (6-bit address) for 32 PHYs (2 Devices, each with up to 16 PHYs). The devices in extended mode are standard compliant (5-bit address).
- Internal Rate:
  - 4 internal rates groups.
  - The internal rate should be set either to PPS mode or to external rate mode.
- Tx scheduling:
  - SPHY: Internal rate shapes the actual transmission rate on the POS-PHY bus. Packets are prefetched to the Virtual TxFIFO.
  - Transmit ordering arbitration is handled in a fixed or round robin (for fairness) priority scheme among pending qualified PHYs within the same device.
  - 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 Packet Transfer
  - Interleave packet transfer to multiple ports.
  - Automatic Odd or Even data parity generation
- Rx selection (two priority levels decision):
  - Within a device, round robin selection among all PHYs of the same priority.
  - 2 priority levels per PHY (reduce worst case round trip latency time for high priority PHY).
- Rx Packet transfer
  - Performs segmentation of received packets from multiple ports.
  - Segment size programmable in UPHEC register.
  - Overrun detection if FIFO is full.
  - Detection of framing errors: Double or late SOP in segment, no RVAL.
  - Switching when RVAL negated.
  - Optional data parity check (Even or Odd parity are supported)
  - Back to back packet transfer with two dead cycles switch time.
- Polling
  - Polling is periodic and fair.
  - Independently programmable last PHY address for Rx and for Tx.
  - Direct status polling is assumed for SPHY device.
  - Internal Rate mechanism to pace polling on transmit MPHY or en-queuing in SPHY
  - Maskable polling per port (rx, tx or both)

33.2.5 Internal Rate Features

Internal rate paces the Transmit FIFO de-queuing rate for SPHY and the en-queuing 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.

- Supprt external rate (internal rate timers disabled)
- Fastest PHY to Slowest PHY transmit rate ratio of 32768 (for example, the fastest PHY could be set to 622 Mbps and the slowest PHY to19 Kbps).
- Configurable expiration counter threshold (cell/packet burst size) per device (1 to 16). When setting to 1 (POS PPS 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 for each device. 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.

33.3 Modes of Operation

33.3.1 UTOPIA Mode

The QUICC Engine ATM controller interfaces with a PHY device through the UPC UTOPIA interface. The QUICC Engine block supports UTOPIA level 2 for both master and slave modes.

33.3.1.1 UTOPIA Master Mode

As master, the UPC controls one UL2 bus with up to 4 devices, and can route each to any one of 4 UCCs. Each device can operate as SPHY or MPHY.

1. 16 or less, configured in UPRP[TIREC].

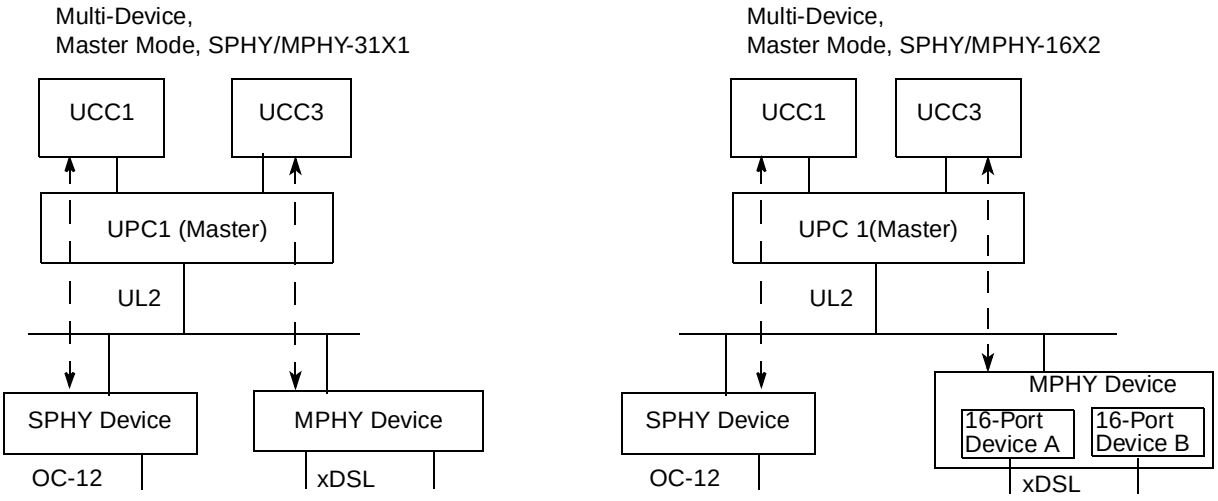


Figure 33-2. Example: Multi-Device Configurations in Master Mode

33.3.1.1.1 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.

In multi-device or single-device modes, single-Clav polling is applied. There is one Clav and one Enb signal per device for up to 4 devices in default MPHY mode or 4 device pairs in 6-bit addressing MPHY mode. During the polling phase, all devices sample the address bus and each device uses its Clav to indicate the status of the corresponding PHY. During the selection phase, the selected PHY address appears on the common address bus, but only the selected device's Enb signal is asserted.

33.3.1.1.2 Default Addressing MPHY Mode

Up to 31 PHYs can be supported on a device, for a total of 124 PHYs in default MPHY mode (total of 124 PHYs). The routing method for default MPHY mode devices is displayed in Figure 33-3.

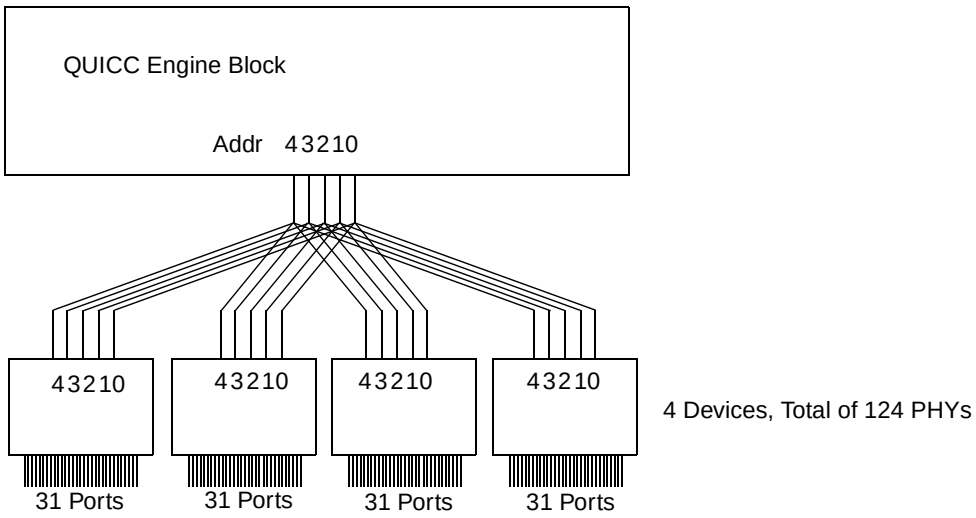


Figure 33-3. Default MPHY Devices address bus routing

33.3.1.1.3 6-bit Addressing MPHY Mode

Up to 32 PHYs can be supported on a device pair of 16-ports physical devices (termed 1A, 1B; 2A, 2B; 3A, 3B; 4A, 4B) in 6-bit addressing MPHY mode (total of 128 PHYs). An extra address bus msb is driven to support two devices on each enb/clav pairs, with address range 0-0xF per device. The routing method for 6-bit addressing MPHY mode devices is displayed in Figure 33-4.

33.3.1.2 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 4 UCCs. The outputs/inputs of devices 2,3,4 (such as Enb[2:4]) are inactive. The UPC use direct status polling for SPHY, and 1-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.

33.3.1.2.1 UTOPIA Slave MPHY Operation

The QUICC Engine block supports one slave in MPHY configuration. The slave address in MPHY mode is configurable (independent address for Tx slave and Rx slave).

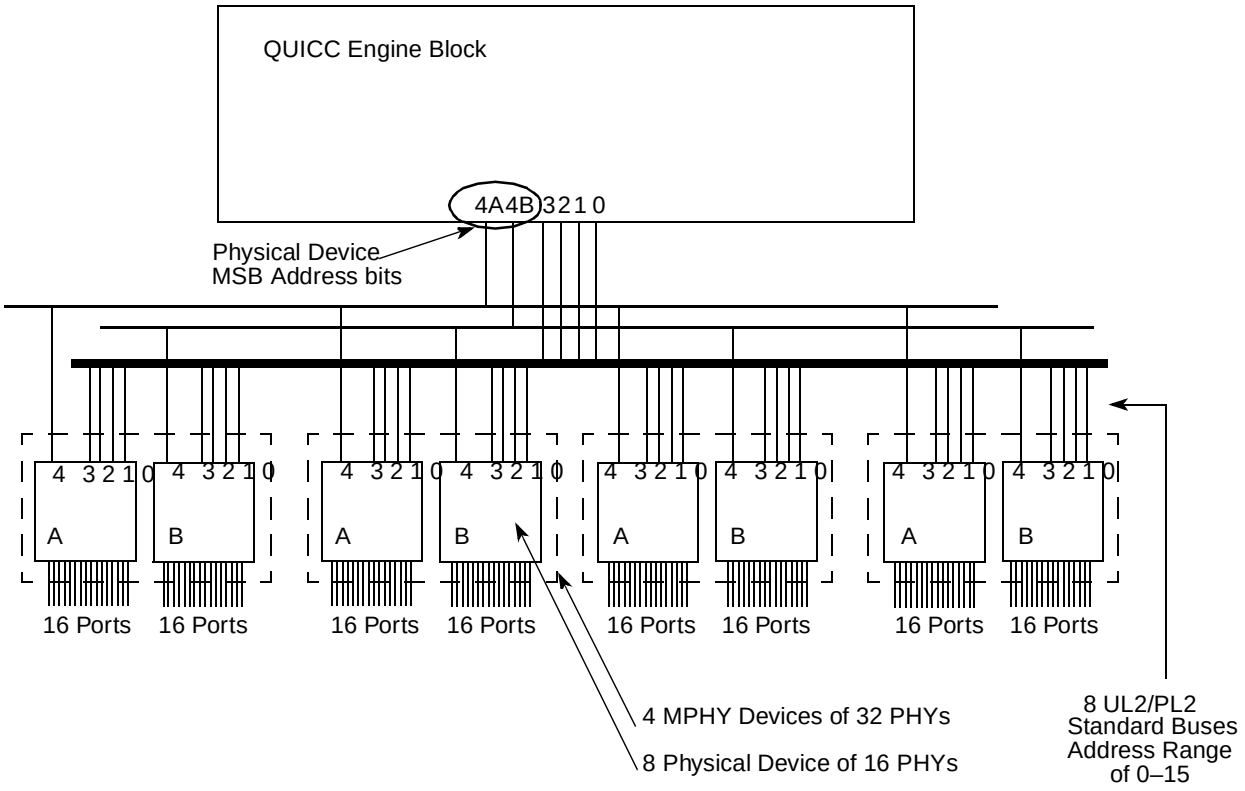


Figure 33-4. 6-bit Addressing MPHY Devices Address Bus Routing

In multiple PHY UTOPIA slave mode, cells are transferred using cell-level handshake as defined by the UTOPIA level-2 standard. The user should write the ATM controller PHY address in UPLPAy[PHY ID].

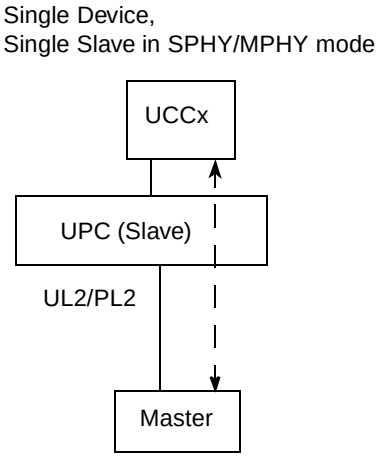


Figure 33-5. SPHY/MPHY Configuration in Slave mode

33.3.2 POS Mode

PL2 works according to POS packet-level handshake.

NOTE

POS supports one device, up to 32 ports in Master mode and does not support internal loopback.

33.3.2.1 POS Master Mode

The UPC as PL2 Master can interleave the transfer of several packets to/from differnt ports by switching. Switching is the use of flow control (pause) to de-select and re-select ports during packet transfer even before the end of the packet. The UPC as Rx Master perform segmentation of packets based on the segment size programmed in the UPHEC register (back to back transfer is optional for SPHY devices). The default and 6-bit addressing modes are displayed in Figure 33-3, and Figure 33-4.

33.3.2.1.1 Internal rate settings for POS Tx (PPS mode)

For POS, the UPC internal rate registers should be configured for the desired time interval between start of packet and the start of the next packet. See section Section 33.3.3, “Transmit Internal/External Rate Modes. For MPHY, the internal rate expiration counter has to be programmed to 1 (UPRPx[TIREC]=1). For single PHY, it is also possible to use external rate (UPRPx[TIREC]=0).

33.3.2.2 POS Slave Mode

The UPC can function as one PL2 slave, in SPHY or MPHY mode. The slave is using only device 1 and it can be routed to one of any of the 4 UCCs. The outputs/inputs of devices 2,3,4 (such as Enb[2:4]) are inactive. The slave address in MPHY mode is configurable in UPLPAy[PHY ID].

The UPC Packet availale indications are direct status packet avaibale (DRPA/DTPA) for SPHY, and polled status packet available (PTPA/PRPA) for PHYs in MPHY system.

In Rx Slave mode, PTPA and STPA signals logic is based on the DCS’s Rx FIFO threshold. The POS Rx FIFO threshold is 64 bytes. Once the FIFO has less than 64 bytes of empty space, the STPA signal is deasserted.

As Slave, the UPC force the Master to do segmentation of the packets. The UPC as Tx Slave (Rx Slave from perspective of POS Rx Master) negate PRPA after selection, so the Master is required to end the selection not beyond the programmed segment size. Likewise, the UPC as Rx Slave (Tx Slave from perspective of POS Tx Master) negate STPA and PTPA after selection so the Master is required to end the selection not beyond the programmed segment size.

33.3.3 Transmit Internal/External Rate Modes

The UPC controller supports the following two rate modes:

- **External Rate (ER) mode** (UPRPx[TIREC]=0) - The total transmission rate is determined by the PHY transmission rate. External rate mode is the default configuration, and it is useful for users who want to have 8260-like behavior of external rate ATM without idle cells, and for POS with variable length packets.
- **Internal rate 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 UPTIRRx. The UPTIRRx includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer expires.

33.3.3.1 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 micro-sec)
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

33.3.3.2 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.

33.3.4 SPHY System Design

This section applies to master mode only.

33.3.4.1 Single Device System

When the system includes only one SPHY device, clear the MultiPHY mode for the UCCs routed to the SPHY in UPUC register (UPUCx[TMP]=0). For receive, clear MultiPHY mode in UPDCy[RMP]. The Clav must be a direct status (always valid). If the address bus is used, refer to Section 33.3.4.2, “Multiple Devices System.”

33.3.4.2 Multiple Devices System

If multiple devices are connected to the UPC, each one that is a SPHY device should be routed to a dedicated UCC and configured as SPHY in the UPC (UPUCx[TMP] = 0). Because there are multiple devices on the UTOPIA/POS-PHY buses, the device must tri-state its outputs (except Clav/PxPA) when not selected. The Clav/PxPA outputs must not be tri-stated. Therefore, some some PHYs must be configured for a multi-PHY bus with direct status indication. The user should pull down the Rx and Tx address bus of the PHY, configure the device to tri-state its outputs when not selected, configure the PHY address to zero, and configure a direct-status Clav indication. The last step (direct-status) may or may not be required because the address lines are static. Consult the PHY data sheet. The Clav should be always active, and the PHY is selected for Rx or Tx cell transfer by assertion of the respective Enb signal.

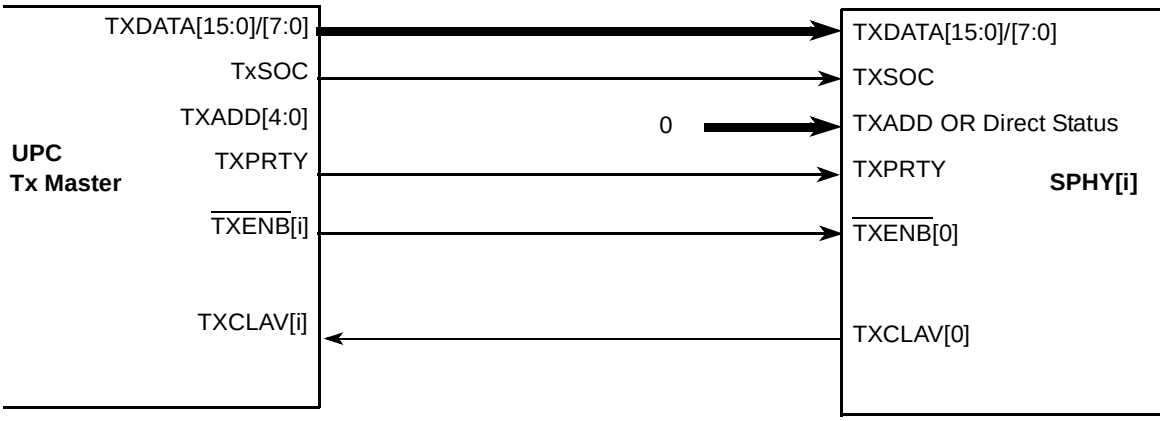


Figure 33-6. UPC with Tx UTOPIA SPHY System

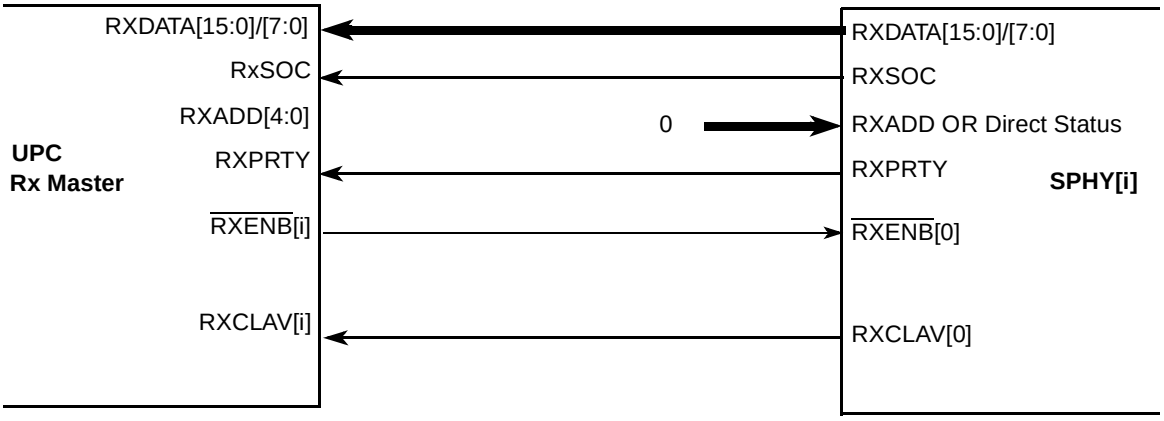


Figure 33-7. UPC with Rx UTOPIA SPHY System

33.3.4.3 Device with Direct Status Polling Indication

The UPC supports direct status polling with simple external glue logic. The user can route a single device with up to 4 internal ports and direct status polling indication to 4 UCCs and handle each port as a SPHY logical device. The respective Rx and Tx Enb signals must be ANDed on the PCB to form a single logical Enb signal that is connected to the device.

33.3.5 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 33-1. 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: The QUICC Engine block de-queues FIFO as a leaky bucket at internal rate. Single Scheduler for all channels. Multi Ports: The QUICC Engine 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.

Table 33-1. SPHY related Modes (continued)

Field Setting	Description	Meaning
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

33.3.5.1 Single Port

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 will attempt to transmit a cell/packet on each expiration of the internal rate counter, and if a time lag has been accumulated it will attempt 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 inter-cell/packet arrival time.

33.3.5.2 Multiple Port

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 channels with similar QOS are bundled together, especially if the channels in each bundle have the same order bit-rate, but channels of different bundles have very high bit rate ratios.

33.3.6 Loop-Back Mode

The UPC supports loop-back mode. In this mode, the Rx and Tx UTOPIA signals are cross-connected internally. Output pins are driven; input pins are ignored.

In loop-back mode, the transmitter and receiver must operate in complementary modes. If the transmitter is master, the receiver is slave and vice versa.

Device1 has to be configured as the Tx or Rx slave, and can be routed to any one of the 4 UCCs. On the opposite direction, the master can be configured to work with more than one device. Each device on the master side can be routed to any of the 4 UCCs. In Loop-Back mode the devices Enb outputs on the master side are shorted to device1 Enb input at the slave side. Device1 Clav/PxPA outputs on the slave side are split, and they become the input to all devices on the master side. Note that only one slave address can be configured in the MPHY system. However, there is a restriction. When Loop-back mode is set and the slave is configured to MPHY mode, all UCCs on the master side must also be configured to MPHY mode. This restriction is due to the multiplexing of the Clav/PxPA signals between the master and the slave (see Figure 33-8, Figure 33-9, Figure 33-10, and Figure 33-11). If the slave is configured to SPHY mode, there

is no similar restriction on the master side, and UCCs on the master side can be configured to either MPHY or SPHY mode.

The master can be configured to work in 6-bit addressing mode. The extra msb address signal is unconnected in loop-back mode, and the slave responds to an address driven on ADDR[4:0] only. The slave address should be between 0–15 if extended address mode is selected in a loop-back system.

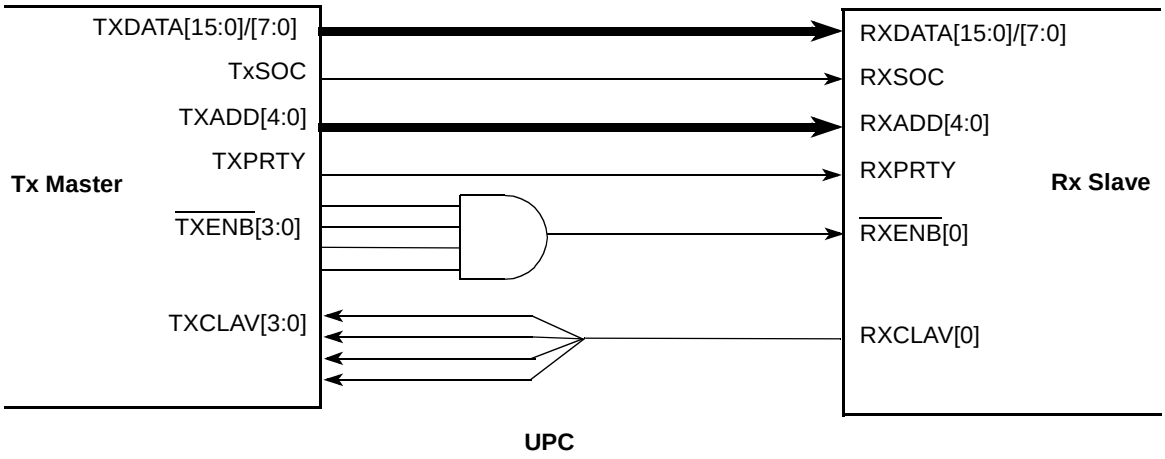


Figure 33-8. UTOPIA Tx Master to Rx Slave Internal Loop-Back System

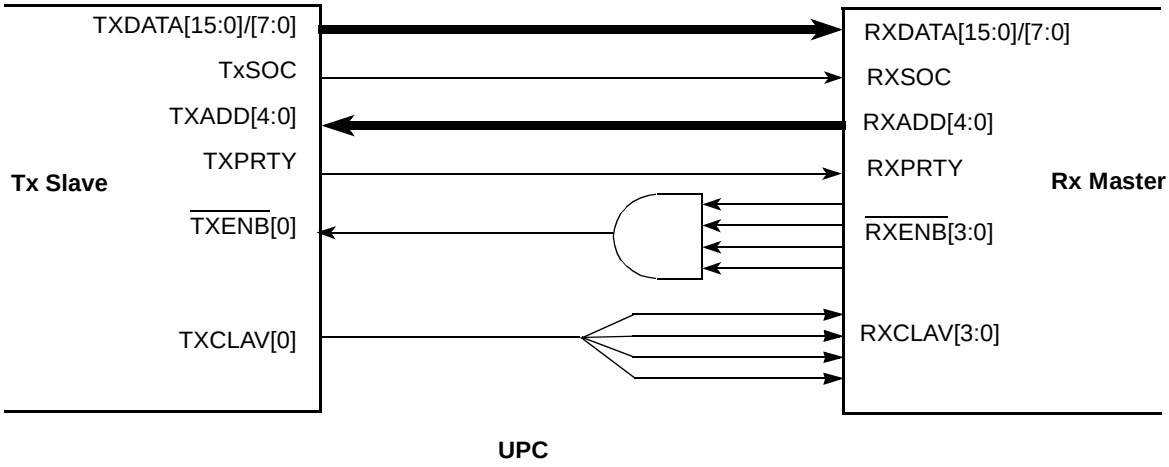


Figure 33-9. UTOPIA Tx Slave to Rx Master Internal Loop-Back System

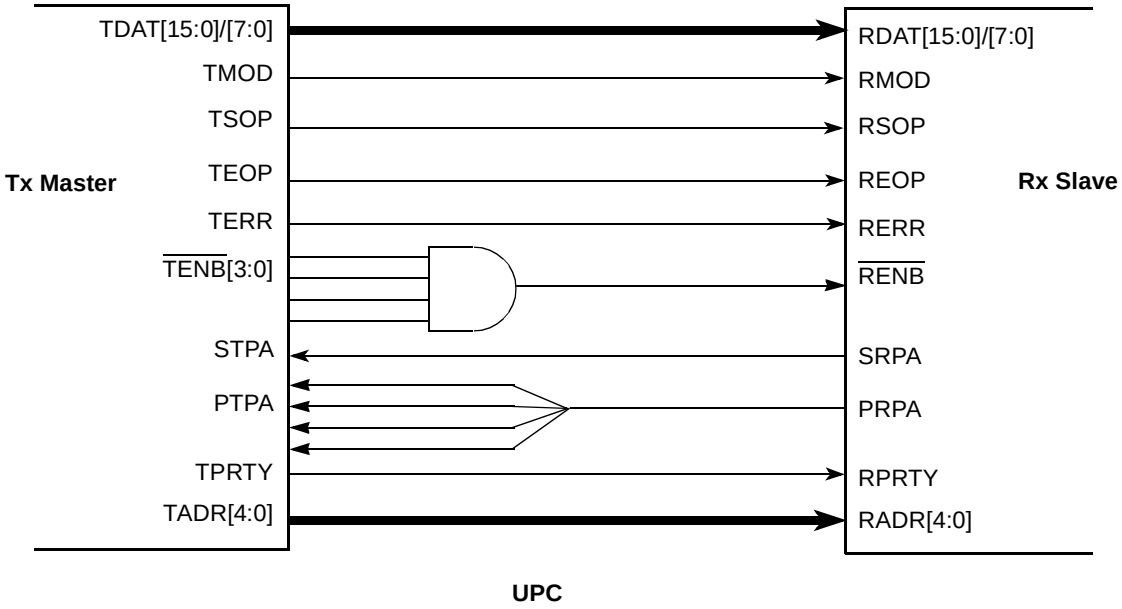


Figure 33-10. POS Tx Master to Rx Slave Internal Loop-Back System

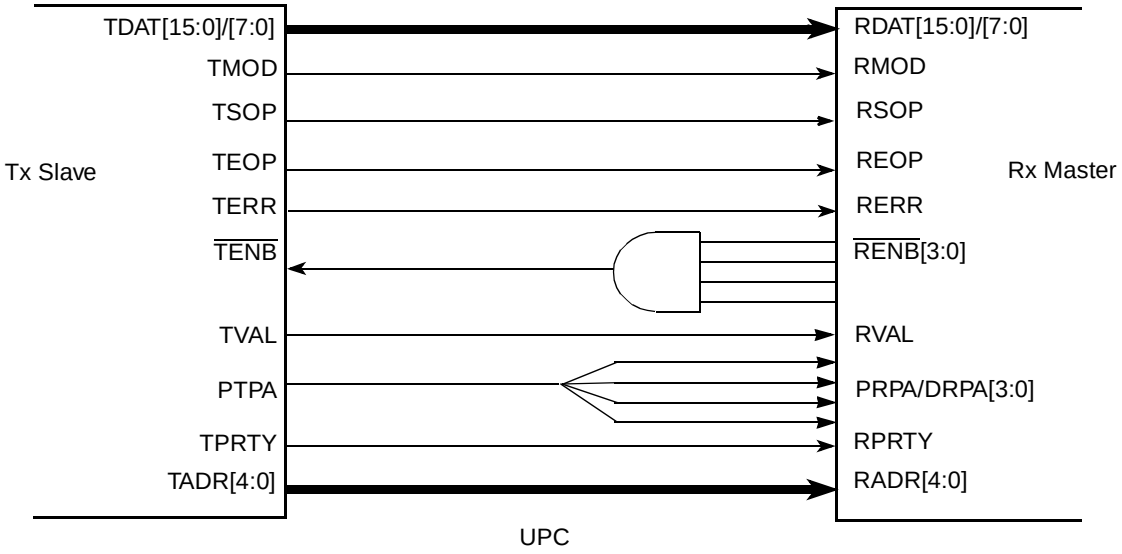


Figure 33-11. POS Tx Slave to Rx Master Internal Loop-Back System

33.3.6.1 Loop-Back Mode Programming Model Example

The example discussed in this section consists of a system with a Tx Master connected to three devices (1, 2, 3) and an Rx slave on device1. The Tx master devices 1 and 2 are routed to UCC1, and device 3 is routed to UCC2. The Rx slave device1 is routed to UCC2. This system can check several features of the UPC. If an Rx slave device is configured as an SPHY, it can respond to all addresses and simulate a full MPHY system. Therefore, configuring Tx Device 3 as a fast SPHY and devices 1 and 2 as MPHY allows features such as IR and B2B to be examined. The programing model for the system is shown in Table 33-2.

Table 33-2. Configurations for Loop-Back Example

Configuration	Bit Value	Description
UPC configurations	UPGCR[DIAG] = 01	Loop-Back 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
	UPUC2 [TMP] = 0	SPHY mode
	UPUC2[TMP] = 0	SPHY mode
	UPUC2[TB2B] = 1	UCC2 Tx B2B enabled
Device Configurations	UPDC1[Device Tx Routing] = 00	Route Tx device 1 to UCC1
	UPDC2 [Device Tx Routing] = 00	Route Tx device 2 to UCC1
	UPDC3[Device Tx Routing] = 01	Route Tx device 3 to UCC2
	UPDC1[Device Rx Routing] = 01	Route Rx device 1 to UCC3

33.4 External Signal Descriptions

NOTE: Slave Mode Signal Naming

Users familiar with the UTOPIA interface of the CPM in MPC82xx and MPC85xx should know that the external signal naming convention of the QUICC Engine block follows the UTOPIA standard and therefore have different naming in master and in slave modes. The naming convention in the CPM retains the master mode signal naming for slave mode. For example, the QUICC Engine block transmit SOC in slave mode is named RSOC but the CPM transmit SOC in slave mode is named TSOC.

In the QUICC Engine block, users should connect signals between master and slave by name. In the preceding example, the user should connect the master TSOC with the QUICC Engine block TSOC.

33.4.1 Overview

This section describes the UPC external signals. The description is divided into UTOPIA mode and POS mode signal groups. There are several bus configurations, depending on the implemented protocol (UTOPIA/POS), the polling method, number of devices in the UTOPIA/POS configuration, and the data bus width. For example, one UTOPIA device configuration requires a total of 36 I/O ports for 8-bit mode and 52 bits in 16-bit mode. Multi devices configuration requires additional 4 control signals (Clav+Enb, for Tx and Rx) for each additional device. 6-bit addressing MPHY mode requires a total of 50 I/O ports for 8-bit mode and 66 bits in 16-bit mode. Table 33-3 summarizes all possible I/O pins assignments combinations.

Table 33-3. UCC UTOPIA/POS I/O Pin Count

	UTOPIA 31 PHYs	UTOPIA 124 PHYs	POS 31 PHYs	POS 124 PHYs	UTOPIA 128 PHYs
8 bit Data	16	16	16	16	16
16 bit Data	32	32	32	32	32
parity	2	2	2	2	2
Address	10	10	10	10	12
Clav/xPTA	2	8	4[1]	10	8
Clock	2	2	2	2	2
Framing	2	2	8[2]	8	2
Enable	2	8	2[3]	8	8
Total for 8 bit	36	48	42[4]	54	50
Total for 16 bit	52	64	60	72	66

1 PTPA, STPA, RVAL, PRPA
2 TSOP, TEOP, RSOP, REOP, TMOD, RMOD, TERR, RERR
3 TENB, RENB
4 Non standard solution. RMOD, TMOD not needed

33.4.1.1 UTOPIA Interface Master Mode

UTOPIA master signals are shown in Figure 33-12.

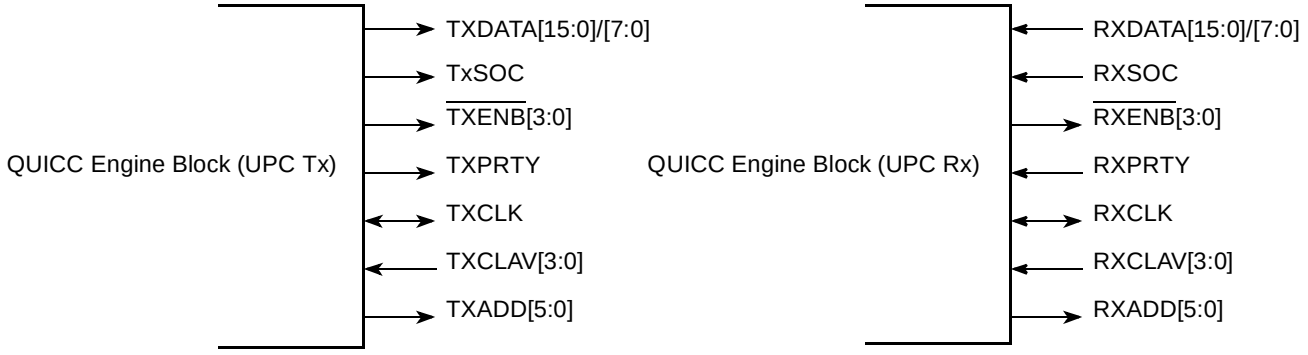


Figure 33-12. UTOPIA Master Mode Signals

Table 33-4 lists the I/O pins in UTOPIA Master mode.

Table 33-4. UTOPIA Master mode Signal Properties

Name	Function	I/O	Tri-State[1]	Pull[2]
TxDATA[15-0]/[7-0]	Transmit data from link layer to PHY	O	Y	Y
TxSOC	Transmit start of cell	O	Y	Y

Table 33-4. UTOPIA Master mode Signal Properties (continued)

Name	Function	I/O	Tri-State[1]	Pull[2]
$\overline{\text{TxENB}}$[3-0]	Transmit enable	O	N	N
TxCLAV[3-0]	Transmit cell available	I		Down
TxPRTY	Transmit parity	O	Y	Y
TxCLK	Transmit clock (up to 50MHz)	I		
TxCLKO	Transmit clock out (up to 50MHz). Output to PHY.	O	N	N
TxADD[5-0]	Transmit address, TxADD[5] is the extra msb (TxAddr[4] for device B)	O	N	N
RxDATA[15-0]/[7-0]	Receive data from PHY to link layer	I		Y
RxSOC	Receive start of cell	I		Down
$\overline{\text{RxENB}}$[3-0]	Receive enable	O	N	N
RxCLAV[3-0]	Receive cell available	I		Down
RxPRTY	Receive parity	I		Y
RxCLK	Receive clock (up to 50MHz)	I/O	N	N
RxADD[5-0]	Receive address, RxADD[5] is the extra msb (RxAddr[4] for device B)	O	N	N

1 Outputs tri-State by UPC as UL2 Master.

2 Pull resistors are required.

33.4.1.2 UTOPIA Interface Slave Mode

UTOPIA slave signals are shown in Figure 33-13.

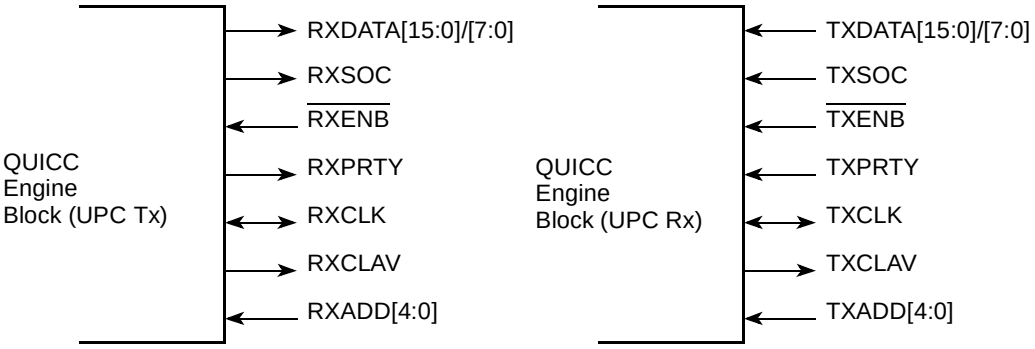


Figure 33-13. UTOPIA Slave Mode Signals

Table 33-5 lists the I/O pins in UTOPIA Slave mode.

Table 33-5. UTOPIA Slave Mode Signal Properties

Name	Function	I/O	Tri-State[1]
TxDATA[15–0]/ [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	Y
TxPRTY	Transmit parity	I	
TxCLK	Transmit clock at 50MHz	I	
TxCLKO	Transmit clock at 50MHz	O	N
TxADD[4–0]	Transmit address	I	
RxDATA[15–0]/ [7–0]	Receive data from link layer to PHY	O	Y
RxSOC	Receive start of cell	O	Y
$\overline{\text{RxENB}}$[0]	Receive enable	I	
RxCLAV[0]	Receive cell available	O	Y
RxPRTY	Receive parity	O	Y
RxCLK	Receive clock at 50MHz	I/O	N
RxADD[4–0]	Receive address	I	

1 Outputs tri-State by UPC as UL2 Slave.

33.4.1.3 POS Interface Master Mode

POS master signals are shown in Figure 33-14.

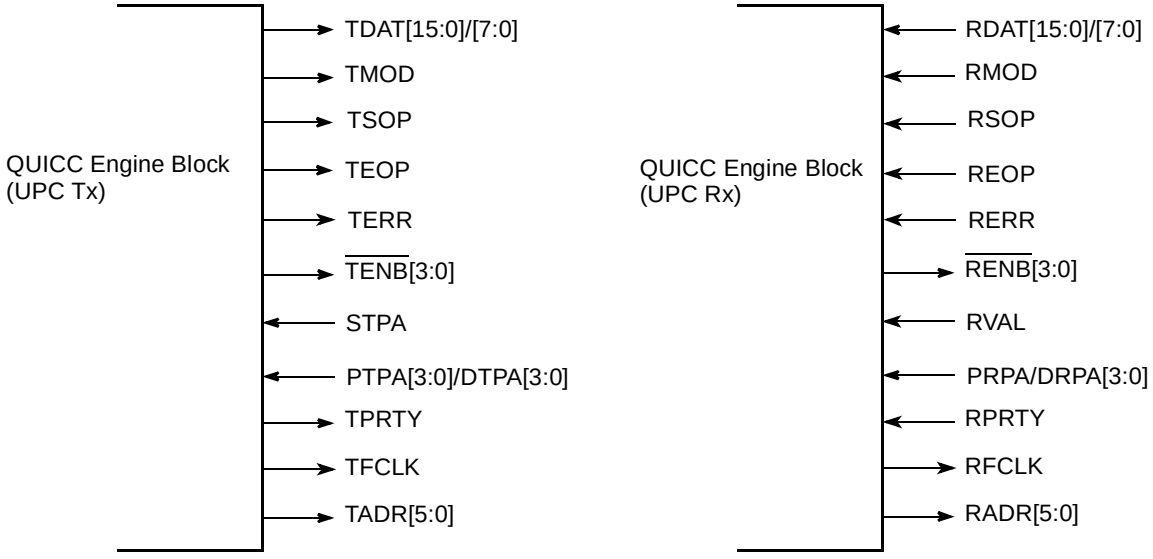


Figure 33-14. POS Master Mode Signals

Table 33-6 lists the I/O pins in POS Master mode.

Table 33-6. POS Master mode Signal Properties

Name	Function	I/O	Tri-State[1]	Pull[2]
TDAT[15:0]/[7:0]	Transmit data from link layer to PHY	O	Y	Y
TMOD	Transmit word modulo	O	Y	Down
TSOP	Transmit start of packet	O	Y	Down
TEOP	Transmit end of packet	O	Y	Down
TERR	Transmit error	O	Y	Down
$\overline{\text{TENB}}$[3:0]	Transmit multi-PHY write enable	O	N	N
STPA	Transmit selected PHY packet available	I		Down
PTPA/DTPA[3:0]	MPHY Transmit polled PHY packet available SPHY Transmit direct PHY packet available	I		Down
TPRTY	Transmit parity	O	Y	Y
TFCLK	Transmit clock from link layer to PHY at 50MHz	I		N
TFCLKO	Transmit clock from link layer to PHY at 50MHz	O	N	N
TADR[5:0]	Transmit address, TADR[5] is the extra msb (TADR[4] for device B)	O	N	N
RDAT[15:0]/[7:0]	Receive data from PHY to link layer	I		Y
RMOD	Receive word modulo	I		Down
RSOP	Receive start of packet	I		Down
REOP	Receive end of packet	I		Down
RERR	Receive error	I		Down
$\overline{\text{RENB}}$[3:0]	Receive multi-PHY read enable	O	N	N
RVAL	Receive data valid	I		Down
PRPA/DRPA[3:0]	MPHY Receive polled PHY packet available SPHY Receive direct PHY packet available	I		Down
RPRTY	Receive parity	I		Y
RFCLK	Receive clock from link layer to PHY at 50MHz	O	N	N
RADR[5:0]	Receive address, RADR[5] is the extra msb (RADR[4] for device B)	O	N	N

1 Outputs tri-State by UPC as POS Master

2 Pull resistors are required.

33.4.1.4 POS Interface Slave Mode

POS slave signals are shown in Figure 33-15.

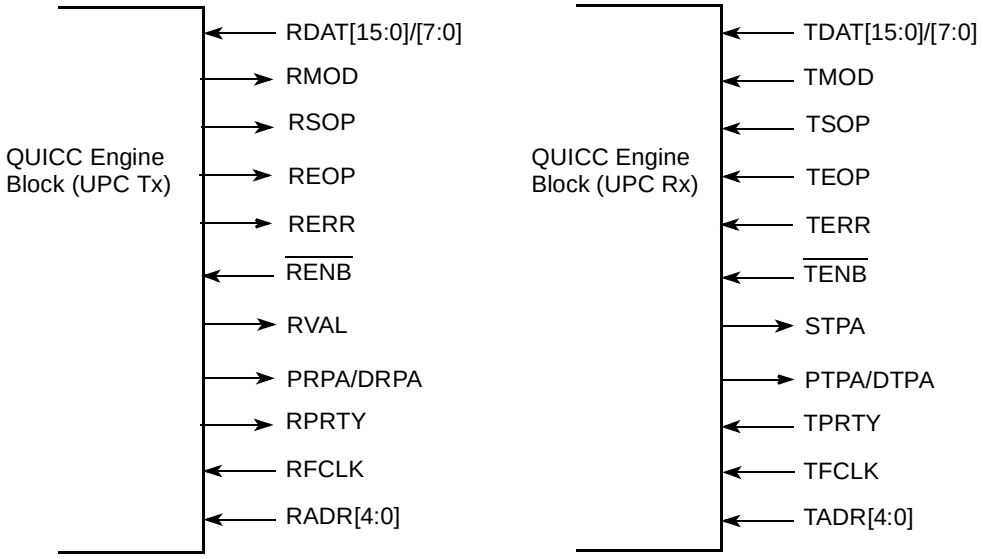


Figure 33-15. POS Slave Mode Signals

Table 33-7 lists the I/O pins in POS Slave mode.

Table 33-7. POS Slave mode Signal Properties

Name	Function	I/O	Tri State[1]
RDAT[15:0]/[7:0]	Receive data from link layer to PHY	O	Y
RMOD	Receive word modulo	O	Y
RSOP	Receive start of packet	O	Y
REOP	Receive end of packet	O	Y
RERR	Receive error	O	Y
$\overline{\text{RENB}}$[0]	Receive multi-PHY write enable	I	
RVAL	Received Data Valid	O	Y
PRPA/DRPA[0]	MPHY Receive polled PHY packet available SPHY Receive direct PHY packet available	O	Y
RPRTY	Receive parity	O	Y
RFCLK	Receive clock from link layer to PHY at 50MHz	I/O	
RADR[4:0]	Receive address	I	
TDAT[15:0]/[7:0]	Transmit data from PHY to link layer	I	
TMOD	Transmit word modulo	I	
TSOP	Transmit start of packet	I	
TEOP	Transmit end of packet	I	
TERR	Transmit error	I	
$\overline{\text{TENB}}$[0]	Transmit multi-PHY read enable	I	
STPA	FIFO has space for additional 64 bytes	O	Y

Table 33-7. POS Slave mode Signal Properties (continued)

Name	Function	I/O	Tri State[1]
PTPA/DTPA[0]	MPHY Transmit polled PHY packet available SPHY Transmit direct PHY packet available	O	Y
TPRTY	Transmit parity	I	
TFCLK	Transmit clock from link layer to PHY at 50MHz	I	
TFCLKO	Transmit clock out from PHY to Link layer at 50MHz	O	
TADR[4:0]	Transmit address	I	

1 Outputs tri-State by UPC as POS Slave

33.5 Memory Map

This section provides a memory map and detailed descriptions of registers.

Table 33-8. 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	R/W	0x0000_0000	2
0xC	UPUC - UCC Configuration	R/W	0x0000_0000	4
0x10	UPDC1 - Device1 Config	R/W	0x0000_0000	4
0x14	UPDC2 - Device2 Config	R/W	0x0000_0000	4
0x18	UPDC3 - Device3 Config	R/W	0x0000_0000	4
0x1C	UPDC4 - Device4 Config	R/W	0x0000_0000	4
0x20	Reserved	R/W	0x0000_0000	1
0x28	Reserved			
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	UPDRS2_H - Device2 Rate Select	R/W	0x0000_0000	4
0x3C	UPDRS2_L - Device2 Rate Select	R/W	0x0000_0000	4
0x40	UPDRS3_H - Device3 Rate Select	R/W	0x0000_0000	4
0x44	UPDRS3_L - Device3 Rate Select	R/W	0x0000_0000	4
0x48	UPDRS4_H - Device4 Rate Select	R/W	0x0000_0000	4
0x4C	UPDRS4_L - Device4 Rate Select	R/W	0x0000_0000	4
0x50	UPDRP1 - Device1 Receive Priority	R/W	0x0000_0000	4

Table 33-8. UPC Register Summary (continued)

Offset[1]	Register	Access	Reset Value	Size (bytes)
0x54	UPDRP2 - Device2 Receive Priority	R/W	0x0000_0000	4
0x58	UPDRP3 - Device3 Receive Priority	R/W	0x0000_0000	4
0x5C	UPDRP4 - Device4 Receive Priority	R/W	0x0000_0000	4
Event Registers				
0x60	UPDE1 - Device1 Event	R/W	0x0000_0000	4
0x64	UPDE2 - Device2 Event	R/W	0x0000_0000	4
0x68	UPDE3 - Device3 Event	R/W	0x0000_0000	4
0x6C	UPDE4 - Device4 Event	R/W	0x0000_0000	4
Internal Rate Registers				
0x70	UPRP1 - Device 1 Internal Rate configuration	R/W	0x0007	2
0x72	UPRP2 - Device 2 Internal Rate configuration	R/W	0x0007	2
0x74	UPRP3 - Device 3 Internal Rate configuration	R/W	0x0007	2
0x76	UPRP4 - Device 4 Internal Rate configuration	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	UPTIRR2_0 - Device2 Transmit Internal Rate 0	R/W	0x0000_0000	2
0x8A	UPTIRR2_1 - Device2 Transmit Internal Rate 1	R/W	0x0000_0000	2
0x8C	UPTIRR2_2 - Device2 Transmit Internal Rate 2	R/W	0x0000_0000	2
0x8E	UPTIRR2_3 - Device2 Transmit Internal Rate 3	R/W	0x0000_0000	2
0x90	UPTIRR3_0 - Device3 Transmit Internal Rate 0	R/W	0x0000_0000	2
0x92	UPTIRR3_1 - Device3 Transmit Internal Rate 1	R/W	0x0000_0000	2
0x94	UPTIRR3_2 - Device3 Transmit Internal Rate 2	R/W	0x0000_0000	2
0x96	UPTIRR3_3 - Device3 Transmit Internal Rate 3	R/W	0x0000_0000	2
0x98	UPTIRR4_0 - Device4 Transmit Internal Rate 0	R/W	0x0000_0000	2
0x9A	UPTIRR4_1 - Device4 Transmit Internal Rate 1	R/W	0x0000_0000	2
0x9C	UPTIRR4_2 - Device4 Transmit Internal Rate 2	R/W	0x0000_0000	2
0x9E	UPTIRR4_3 - Device4 Transmit Internal Rate 3	R/W	0x0000_0000	2
0xA0	UPER1 - Device 1 Port Enable	R/W	0x0000_0000	4
0xA4	UPER2 - Device 2 Port Enable	R/W	0x0000_0000	4
0xA8	UPER3 - Device 3 Port Enable	R/W	0x0000_0000	4
0xAC	UPER4 - Device 4 Port Enable	R/W	0x0000_0000	4
0xB0– 0xFF	Reserved			

1 0x2E00 for UPC1, 0x3E00 for UPC2

33.6 UPC Register Descriptions

33.6.1 UPC General Configuration Register (UPGCR)

Offset:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field: R	Protocol	TMS	RMS	ADDR	Res			DIAG								
W																
Reset	0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field: R																
W																
Reset	0000_0000															

Figure 33-16. UPC General Configuration Register (UPGCR)

The UPGCR is an 8 bit read/write register. Table 33-9 describes the UPGCR fields

Table 33-9. UPGCR Register Field Descriptions

Bits	Name	Description
0	Protocol	0 UL2 1 PL2
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 6-bit addressing MPHY device operation. Two 16-port physical devices (A,B) multiplexed on each Clav/Enb pairs for 32 addresses per Clav/Enb. Parallel polling among 4 pairs in multi device system.
4-6	Reserved	Must be cleared.
7	DIAG[1]	Diagnostic mode 0 Normal mode 1 Loop-back mode In Loop-back, UTOPIA/POS Rx and Tx signals are shorted internally. Output pins are driven, input pins are ignored.
8-31	-	Not in use

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].

33.6.2 UPC Last PHY Address Register (UPLPA)

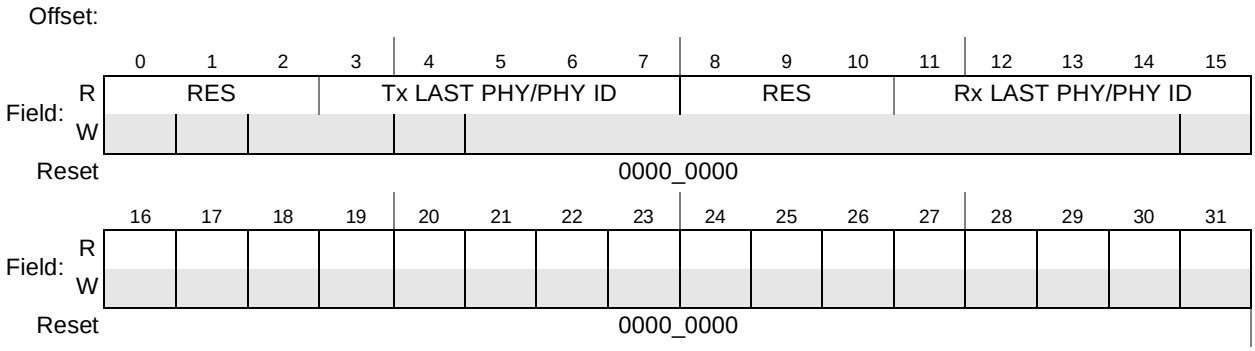


Figure 33-17. UPC Last PHY Address Register (UPLPA)

Table 33-10 describes the UPLPA fields.

Table 33-10. 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. 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: Rx PHY address
16-31	—	Not in use

33.6.3 UPC HEC Register (UPHEC)

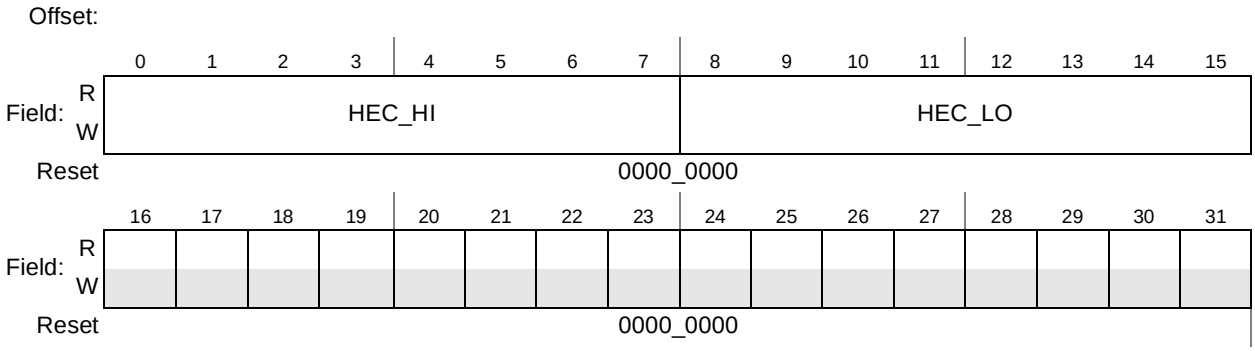


Figure 33-18. UPC HEC Register (UPHEC)

Table 33-11 describes the UPHEC fields.

Table 33-11. 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]. For 16 bit UTOPIA, the value of this field is sent instead of the HEC on TxData[15:8]. 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. For 16 bit UTOPIA, the value of this field is sent with HEC_LO on TxData[7:0]. POS (Rx Master): segment size in bytes (must be an even number, greater than 8). Will attempt to switch to a different port after transfer of a segment with the specified size.
16-31	-	Not in use

33.6.4 UPC UCC Configuration Register (UPUC)

Offset:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field:	R	TMP 0	TSP0	TB2B 0	—	—	—	—	—	TMP 1	TSP1	TB2B 1	—	—	—	—	—
	W																
Reset		0000_0000															

		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field:	R	TMP 2	TSP2	TB2B 2	—	—	—	—	—	TMP 3	TSP3	TB2B 3	—	—	—	—	—
	W																
Reset		0000_0000															

Figure 33-19. UPC UCC Configuration Register (UPUC)

Table 33-12 describes the UPUCx fields.

Table 33-12. UPUC Register Field Descriptions

Bits	Name [1]	Description
0, 8, 16, 24	TMP_x	Transmit Multiple PHY mode. This mode should be set when several devices or a device with several PHYs are 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 tri-state outputs) 1 MPHY system (Output Clav is a response for polling, Tri-state outputs)
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. Can be used in order to enable a PHY back to back (FIFO emptying mode), priorities a group of MPHY devices on the UCC or to simulate Ethernet like operation in POS. See Section 33.7.2, “Transmit and Receive Priority among devices.”
3-7, 11-15, 19-23, 27-31	Reserved	Reserved, should be cleared.

[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.

33.6.5 UPC Device X Configuration Register (UPDCx)

33.6.5.1 UPDCx in ATM Protocol

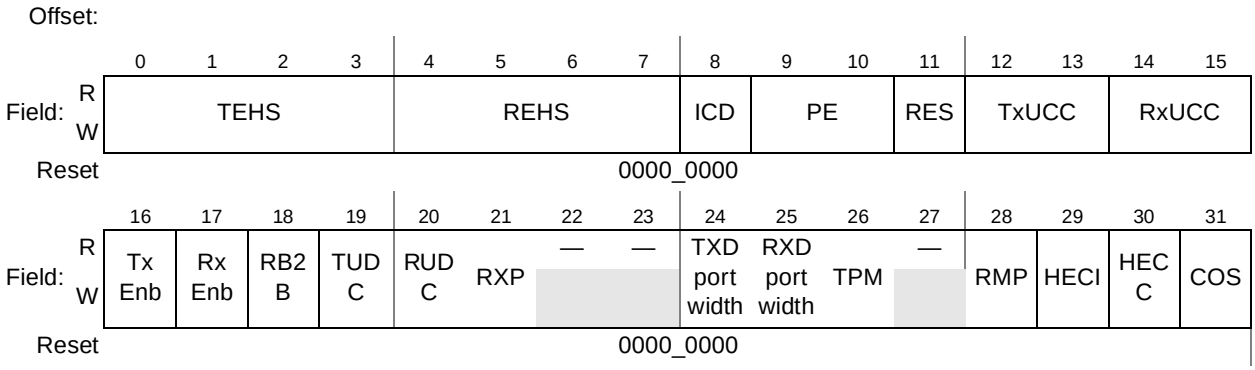


Figure 33-20. UPC Device X Configuration Register in ATM protocol (UPDCx)

Table 33-13 describes the UPDCx fields.

Table 33-13. 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	—	Reserved. Must be cleared
12-13	TxUCC	00 UCC1 (UPC1) or UCC2 (UPC2) 01 UCC3 (UPC1) or UCC4 (UPC2) 10 UCC5 (UPC1) or UCC6 (UPC2) 11 UCC7 (UPC1) or UCC8 (UPC2)

Table 33-13. UPDCx in ATM Protocol Field Descriptions (continued)

Bits	Name	Description
14-15	RxUCC	00 UCC1 (UPC1) or UCC2 (UPC2) 01 UCC3 (UPC1) or UCC4 (UPC2) 10 UCC5 (UPC1) or UCC6 (UPC2) 11 UCC7 (UPC1) or UCC8 (UPC2)
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 an SPHY, or to prioritize a device. See Section 33.7.2, “Transmit and Receive Priority among devices.”
19	TUDC	Transmit user-defined cells 0 Regular 53-byte cells. 1 User-defined cells.
20	RUDC	Receive user-defined cells 0 Regular 53-byte cells. 1 User-defined cells.
21	RXP	Receive parity check. 0 Check Rx parity line. 1 Do not check Rx parity line.
22–23	—	Reserved
24	TXD port width	Transmit data bus width Device/SPHY 0 8-bit data bus width 1 16-bit data bus width Note: For POS 16-bit must be used.
25	RXD port width	Receive data bus width Device/SPHY 0 8-bit data bus width 1 16-bit data bus width Note: For POS 16-bit must b used.
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

Table 33-13. UPDCx in ATM Protocol Field Descriptions (continued)

Bits	Name	Description
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

1 Discard Rx Idle Cell

2 Either direct status or the address lines of the PHY are hard wired fixed to the PHY's 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)

33.6.5.2

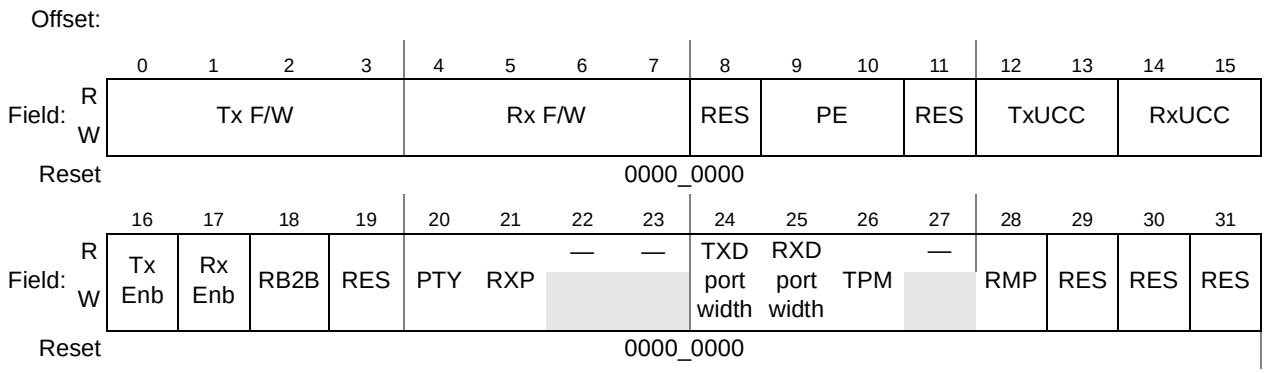


Figure 33-21. UPC Device X Configuration Register in POS protocol (UPDCx)

33.6.5.3 UPC Device X Internal Rate Configuration Register (UPRPx)

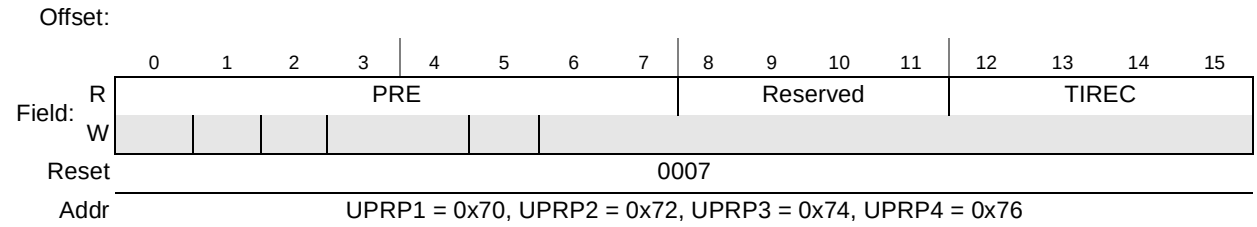


Figure 33-22. Device X Internal Rate Configuration Register (UPRPx)

Table 33-14 describes the UPRPx fields.

Table 33-14. UPRP Field Descriptions

Bits	Name	Description
0-7	PRE	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: Tranmsit rate is limited by Clav, polling and selection priorities. (Idle cells are discarded and not transmitted). 1- PPS mode. No burst and no underrun error indication. Recommended for POS Credit is not expired until the end of packet. Useful also for PON. 2-15 - Internal rate with burst and underrun indication. 7 - Default value (82xx internal rate compatible) This field sets the maximal allowed cell credit for the PHYs. An internal rate expiration counter holds the credit balance per PHY. It is incremented every time the internal rate expires. It is decremented 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 will transmit cells to that PHY. When the credit reaches the value of the TIREC, a transmit underrun event is set in registers UPDEx.

33.6.6 UPC Device X Transmit Internal Rate Register (UPTIRRx)

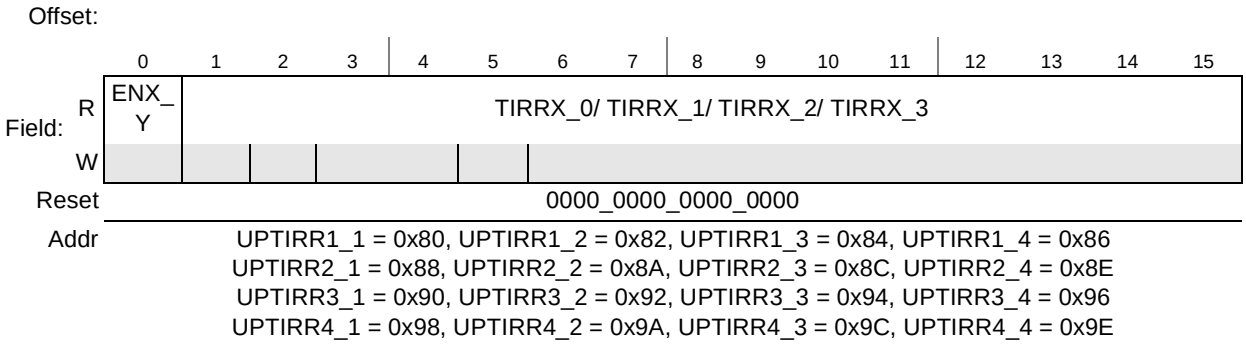


Figure 33-23. UPC Device X Transmit Internal Registers (UPTIRRx_y)

Table 33-15 describes the UPTIRRx fields

Table 33-15. 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

33.6.7 UPC Device X Port Enable Register (UPERx)

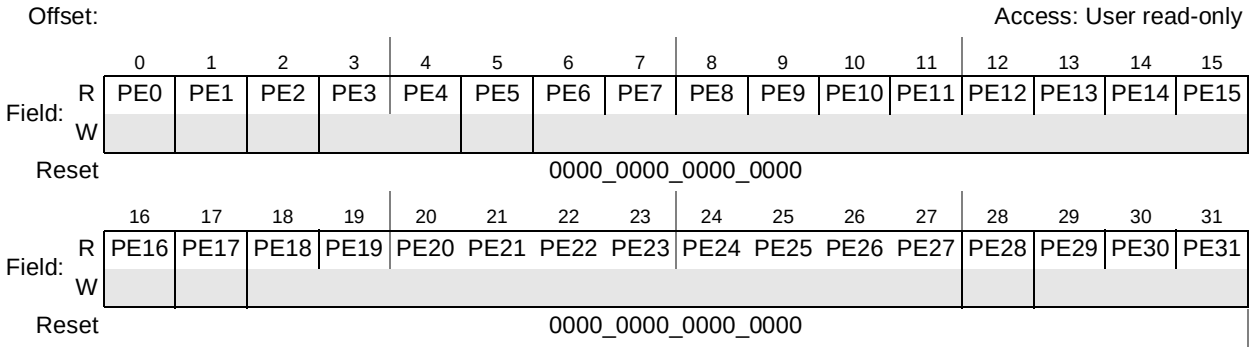


Figure 33-24. UPC Device X Port Enable Register (UPERx)

Table 33-16 describes UPERx fields.

Table 33-16. 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 single end-point (UPUC[TSP] = 0), the meaning of PE0 depends on UPDC[PE]: • UPDC[PE] = 0: Port0 is always enabled and this field have no impact. • UPDC[PE]! = 0: PE0 enable/disables the slave.
1-31	PE<n>	Port <n> Enable. The port enable option depends on the UPDC[PE] field. 0 Port is disabled. Clav/PxPA from this port is internally masked. 1 Port is enabled. For a slave, these fields have no impact except for multi end-point (UPUC[TSP]=1).

33.6.8 UPC Device X Transmit Rate Select Register (UPDRSx)

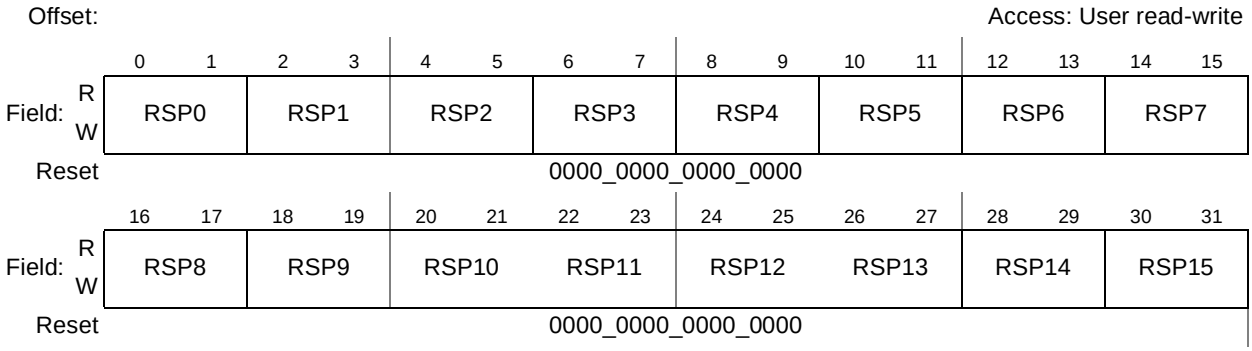


Figure 33-25. UPC Device X Transmit Rate Select, High (UPDRS_Hx)

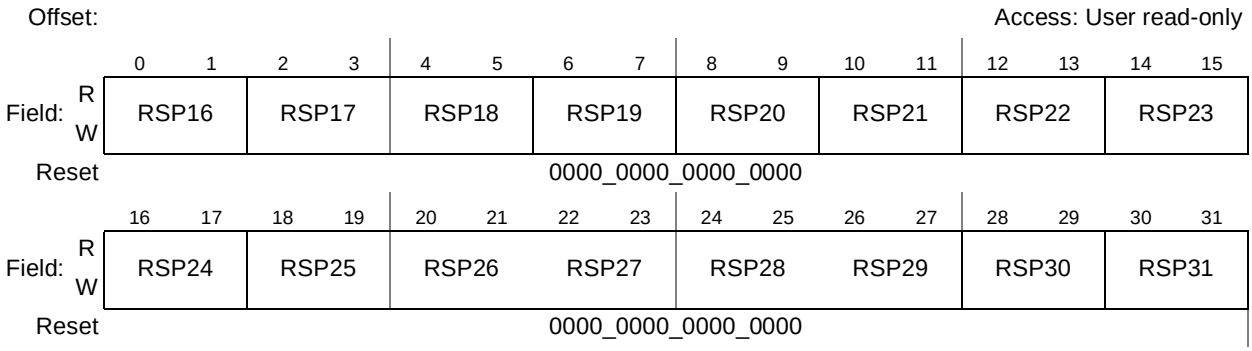


Figure 33-26. UPC Device X Transmit Rate Select, Low (UPDRS_Lx)

Table 33-17 describes the UPDRSx fields.

Table 33-17. 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

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.

33.6.9 UPC Device X Receive Port Priority Register (UPDRPx)

Offset: Access: User read-only

Field:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RPP 0	RPP 1	RPP 2	RPP 3	RPP 4	RPP 5	RPP 6	RPP 7	RPP 8	RPP 9	RPP 10	RPP 11	RPP 12	RPP 13	RPP 14	RPP 15
W																
Reset	0000_0000_0000_0000															

Field:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RPP 16	RPP 17	RPP 18	RPP 19	RPP 20	RPP 21	RPP 22	RPP 23	RPP 24	RPP 25	RPP 26	RPP 27	RPP 28	RPP 29	RPP 30	RPP 31
W																
Reset	0000_0000_0000_0000															

Figure 33-27. UPC Device X Receive Port Priority (UPDRPx), Low Addr

Table 33-18 describes the UPDRPx fields.

Table 33-18. UPDRP Register Field Descriptions

Bits	Name	Description
<n>	RPP<n>	Port <n> Receive Priority[1] 0 RPx0 (High Priority) 1 RPx1 (Low Priority)

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.

33.6.10 UPC Device X Event Register (UPDEx)

Offset: 0xC_0040 Access: User read-write

Field:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EV0	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9	EV10	EV11	EV12	EV13	EV14	EV15
Reset	0000_0000_0000_0000															

Field:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EV16	EV17	EV18	EV19	EV20	EV21	EV22	EV23	EV24	EV25	EV26	EV27	EV28	EV29	EV30	EV31
Reset	0000_0000_0000_0000															

Figure 33-28. UPC Device X Event Register (UPDEx)

Table 33-19 describes UPDEx fields.

Table 33-19. 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 and cleared when the host writes a value of 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 using TIREC value larger then 1. 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.

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.

33.6.11 UPC POS STPA/DTPA Threshold Register (UPSTPA)

In POS Tx Master mode, the UPSTPA register holds the PHY FIFO threshold on which the STPA/DTPA signal is negated. At negation of STPA, the POS master continues the transfer of UPSTPA[STPAT] additional bytes or until last indication. If the last indication is reached, the master ends the packet normally. Otherwise, the master enters an idle state in which it parks the polling address on the current PHY ID and waits for assertion of the PTPA to reselect it. Note that on Rx, the POS master acts in the same manner, and on the negation of RVAL with no EOP, it enters an idle state in which it parks the polling address on the current PHY ID and waits for assertion of the PTPA to reselect it.

As slave, this register is not used. The STPA threshold of the UPC is 64 bytes. This is a number smaller than recommended for a PHY (256 bytes). However, since the VFIFO is usually set for at least 256 bytes, the total FIFO size of the UPC exceeds the recommendation.

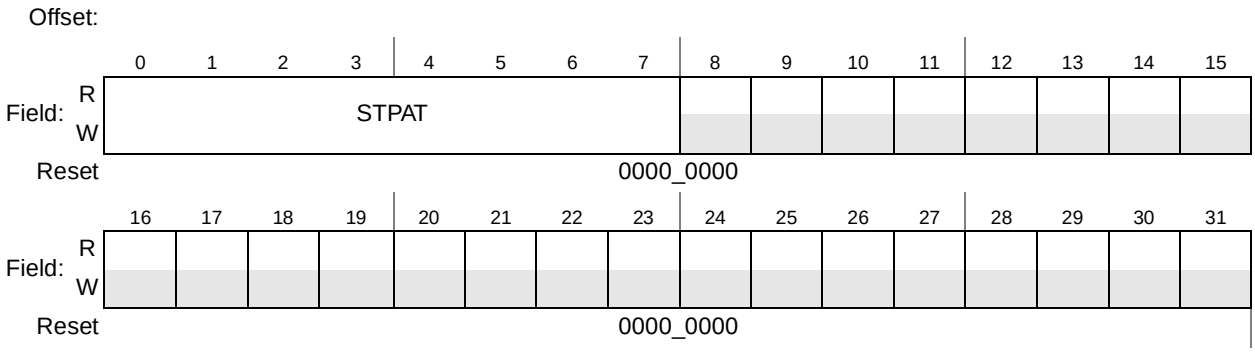


Figure 33-29. UPC STPA Threshold Register (UPSTPA)

Table 33-20 describes UPSTPA fields.

Table 33-20. UPSTPA Register Field Descriptions

Bits	Name	Description
0-7	STPAT	STPA (Selected Transmit Packet Available) Threshold Master mode only: When STPA is asserted, the selected PHY can receive at least <STPAT>+1 bytes or EOP
8-31	—	Not in use

33.6.12 UCCx Event/Mask Registers (UCCEx, UCCMx)

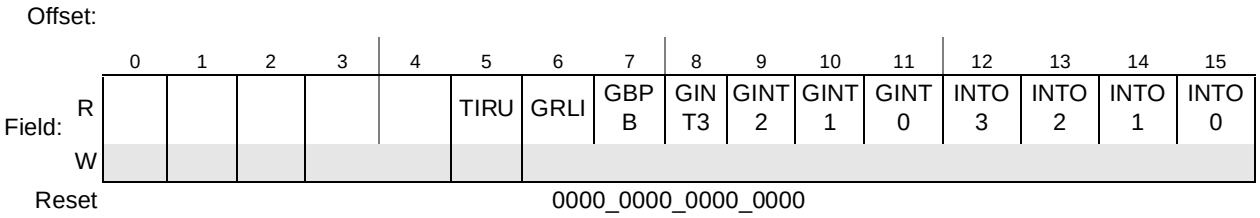


Figure 33-30. UCCEx/UCCMx in ATM protocol

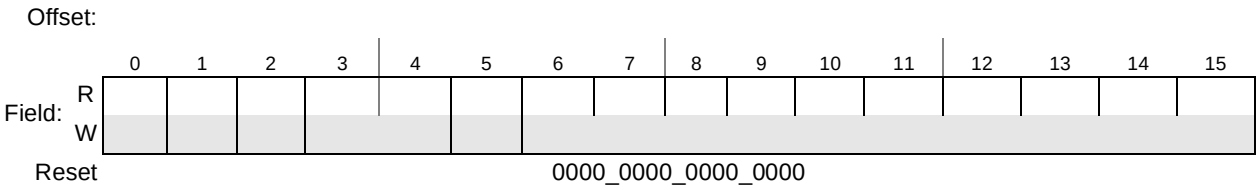


Figure 33-31. UCCEx/UCCMx in POS protocol

33.7 Functional Description

33.7.1 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 BW to fast PHYs over slow PHYs.

Polling address generation is always done 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 to one of 2 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. Set UPDRPx[RPP]=0 for fast PHYs, and UPDRPx[RPP]=1 for slow PHYs.

33.7.2 Transmit and Receive Priority among devices

Selection between devices is always in a round robin and cyclic order, with the exception of back to back selection in a UCC level. This mode can also be described as a PHY TxFIFO fill mode, RxFIFO empty mode.
The user can require that there will be an attempt to do a back to back transfer from a specified UCC. This is required for a master transmitting to a SPHY or a MPHY device with accumulated internal rate that exceeds 50% of the bus bandwidth. There will be an attempt to conduct a burst of cell transfers to the PHY as long as the expiration counter is positive. In such a case the PHY responds with Clav/PxPA and there is a ready cell or packet for the PHY (Tx). For Rx, a back to back transfer will be attempted as long as there is a valid Clav in the cycle following the cell transfer and there is room for one. In general, using this option gives a higher priority to the devices routed to the UCC for which the B2B option is set.

The user can use this option, to allow a fast PHY the required bandwidth. This is done by assigning the devices for which a back to back transfer is required for a UCC different from the rest of the devices, and assigning the back to back transfer option for this UCC. The BW per PHY will be limited by the PHY itself. Otherwise, there is only a round robin based selection among the devices.

33.8 Initialization

The following list is a set of recommendations for efficient management of the bus bandwidth (The UPC is designed to provide sufficient flexibility in allocation of bus bandwidth among the PHYs such that it does not become the bottleneck):

- In a system where one of the PHYs is allocated more then 50 percent of the bus, route that PHY to a separate device on a dedicated UCC in SPHY mode, and set the back to back transfer option for that device. The other PHYs should be routed to other UCCs.
- In a system with one or two fast PHYs (single PHY devices) it is recommended to route each one to a different UCC as SPHY to optimize the use of the transmit FIFOs (each SPHY has a dedicated transmit FIFO). Use the MPHY device option only when assigning more than two PHYs. The reason is that at a given time the FIFO can hold a maximum number of cells equal to the number of PHYs routed to it. Use the Tx internal rate and transmit the back to back transfer option to control the actual rate for both PHYs.
- In general, design the system so that PHYs with slow rates are sharing a device, fast PHYs are assigned to a dedicated FIFO (SPHY mode).
- In POS level 2, it is recommended to use an IR mechanism if the packet size is fixed (cell like packets). That can help managing the bus utilization and it allows transmit underrun errors to be reported. In MPHY mode, set to ER-like mode (UPRP[TIREC]=0) if the packet size can be larger or is variable (max value expiration counter is 1) in order to limit a PHYs credit to only one packet (avoid back to back packet transfers).
- In POS level 2, it is recommended to use B2B option on the transmitter for SPHY Ethernet like devices. For SPHY ATM like devices, use the IR mechanism in the same way as for UL2.

33.9 Glossary[1]

- *Device*. A group of PHYs or ports (could be also a single PHY) controlled by common control signal pair (xADDR[5], 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 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.
- Extended address. A 6-bit xADDR master bus that can be connected to standard 5 bit xADDR slave bus. Up to 4 devices use xADDR[4] as the most significant address line, and another 4 devices use xADDR[5] as the most significant address line (ADDR[MSB]). This option is used for 128 PHY addresses (8 devices of 16 PHYs each).
- *Interleaving*. The term used for handling segments of different packets in sequence.
- *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*. An underrun event of a conceptual transmit jitter buffer. This event is a mismatch of up to 16 cells/packets[2] over any interval 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-Devic*e. Multiple devices on the bus controlled by separate Clav/Enb signals and optionally by a separate xADDR[MSB] (see Extended address).
- *PHY*. A physical bus endpoint, usually with a line connection. A PHY can 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 that 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, UCC7. UPC2 can be routed to UCC2, UCC4, UCC6, UCC8.
- *Segment*. Part of a packet, usually with fixed size unless it contains the end of the packet.
- *Segmentation*. The process of breaking up the transfer of packets to segments. To optimize for bus performance packets are not transferred continuosuly from start to end, but in segments.Segment: Part of a packet, usually with fixed size unless it contains the end of the packet.
- *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 - e.g. an uplink PHY).

1.Applicable to UTOPIA and POS-PHY Level 2.

2.Could be less then 16, the size of the jitter window is configurable.

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.

Switching: The process of de-selecting and re-selecting PHYs for segmentation.

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 34
Multi-Channel Controller (MCC)

34.1 Introduction

The multi-channel controller (MCC) supports up to 256 separate time-division serial channels. It is paired with a serial interface (SI), allowing the MCC to communicate over any of the SI's 8 time-division multiplexed streams (TDM).

Proper programming of the SI and SIRAM is necessary for routing the timeslots within a TDM stream to the appropriate MCC channel at the desired time. Users should be familiar with programming the SI and parallel I/O ports before proceeding.

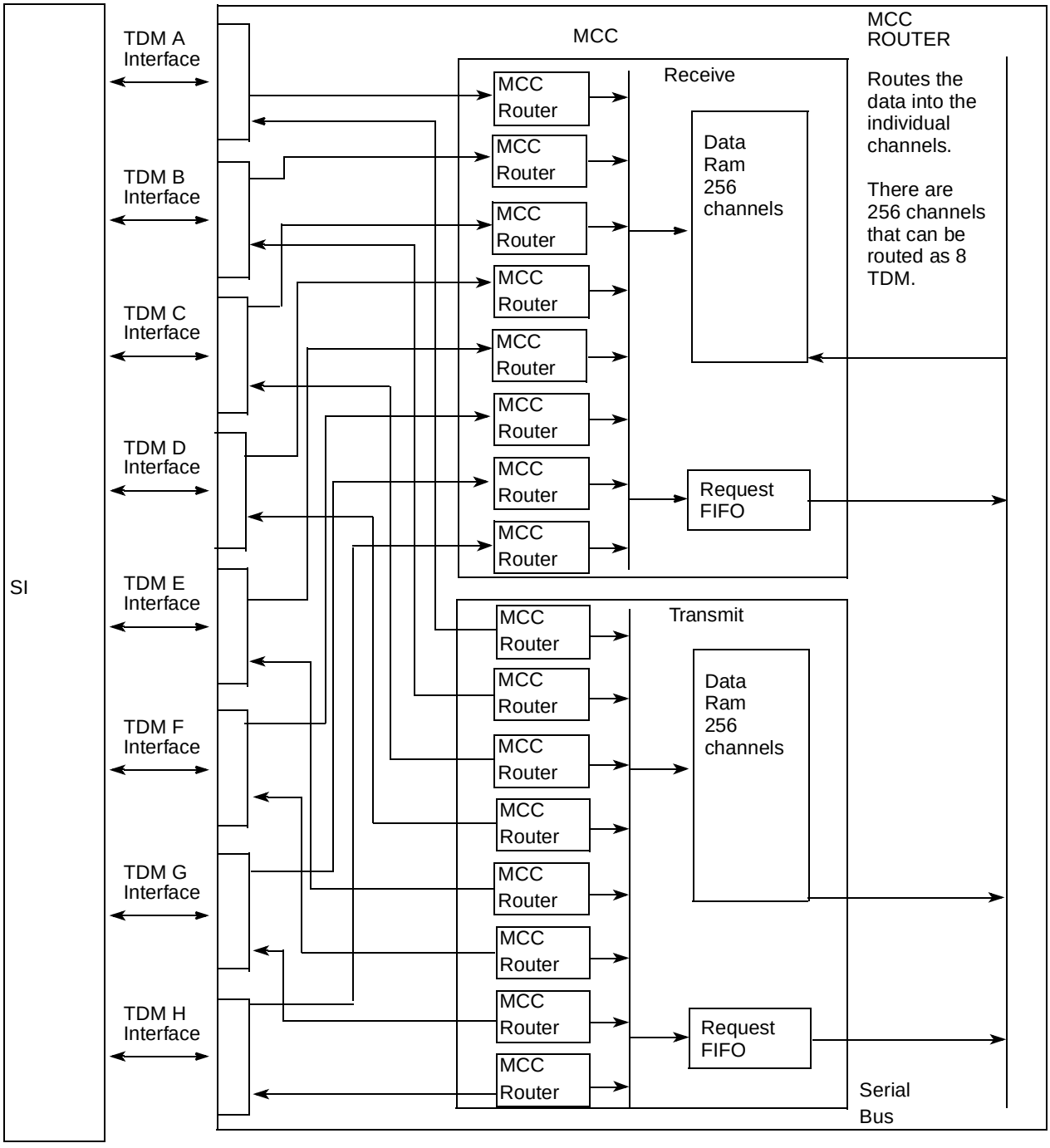


Figure 34-1. MCC Block Diagram

34.1.1 Features

- Supports up to eight TDM lines.
- Up to 256 independent communication channels (up to 8 E1).
- Up to 256 independent communication channels on a single link.

- Independent mapping for receive/transmit.
- Each TDM has 32 channel resolution packers.
- Supports HDLC, transparent, or SS7 protocols on a per-channel basis.
- Supports additional circuit emulation service functionality when used in conjunction with ATM AAL1.
- Supports interworking with AAL0.
- Up to 512 DMA channels (Rx/Tx) with independent buffer descriptor (BD) tables.
- Five interrupt circular tables with programmable size and overflow identification. One for transmit and four for receive.
- Efficient bus usage (no bus usage for inactive channel or for active channels with nothing to transmit).
- Efficient control of the interrupts to the core.
- Supports external BD tables.
- Uses on-chip multi-user RAM (MURAM) for parameter storage.
- Uses 64-bit data transactions for reading and writing data in BDs.
- Supports automatic routing in transparent mode using negative empty polarity.
- Supports inverted data per channel.
- Supports superchannel synchronization in transparent mode (slot synchronization).
- Supports in-line synchronization in transparent mode (synchronization on a pattern of 1 or 2 bytes).
- Provides a processor interrupt on transmit underrun or receive overflow.
- Individual CHANNEL Underrun capability.
- Up to 32 Mbps sustainable aggregated throughput (for HDLC/Transparent) for normal channels and 16-bits super channels. Up to 16 Mbps for other super channels settings.

34.1.2 Comparison with MPC82xx CPM MCC

- Superchannel support—The CPM MCC handled only 1 byte time slots. The QUICC Engine MCC supports 5-, 6-, 7-, 8-, and 16-bit entries.
- The QUICC Engine MCC supports 8 TDMs with 256 channels instead of 4 TDMs with 128 channels.
- The QUICC Engine MCC has the possibility for a channel underrun instead of a global underrun. It means that only the channel that has no data will be affected and not the whole TDM or MCC.
- The QUICC Engine MCC has a programmable Emergency level so the User can alter the service level to the MCC according to the system load it is working in.
- Enhanced debug capabilities.

34.1.3 Modes of Operation

The MCC relies upon its corresponding SI block as its physical interface to a TDM. Once the SI's TDM has been programmed to contain MCC-related timeslots (accomplished via SIRAM programming) and has been enabled, the SI clocks data out of the MCC transmitter, or clocks data into the MCC Receiver. Note

that the SI contains no data buffering, data moving to or from the MCC is clocked directly through the SI, to or from the TDM data lines.

The QUICC Engine block uses a variety of MCC-related data structures to handle that channel's traffic. The QUICC Engine intervention is triggered by requests from the MCC.

The MCC global parameters govern the overall state of the MCC block and also contain some threshold settings and base pointers used by all channels for their operation. There is also one set of channel-specific parameters and channel-extra parameters per MCC channel, containing protocol state information for that channel and pointers to that particular channel's receive and transmit buffer descriptors. These parameter RAM areas are described in more detail in the following sections.

Note that the channel-specific parameter area may be interpreted differently depending on what protocol is being used on that particular channel, whether it is HDLC, transparent, or SS7. If an MCC channel is being used in conjunction with AAL1 CES, there are additional programming model changes that take place. Also, if a TDM is programmed to make use of superchanneled MCC timeslots, a structure called the Superchannel Table is also used. All these parameter RAM areas are described in more detail in the following sections.

34.1.3.1 Transparent

When using this mode, the individual channel in the frame will be in transparent mode. There is an option on the receive side for acquiring synchronization in order to byte or half word alignment.

34.1.3.2 HDLC

When using this mode, the individual channel in the frame will be in HDLC mode. An internal framer will provide standard HDLC features:

- Flag/Abort/Idle generation/detection
- Zero insertion/deletion
- 16-bit CRC-CCITT generation/checking
- Detection of non-octet aligned signal units
- Programmable number of flags between signal units

More details can be found under Section 34.2.2.1, "Channel-Specific HDLC Parameters."

34.1.3.3 Signaling System #7 (SS7)

Based on the HDLC protocol, the signaling system #7 (SS7) protocol is used to manage public service networks. The SS7 protocol operates on signal units (SU), which are analogous to HDLC frames. The physical, data link, and network layer functions of the SS7 protocol are called the message transfer part (MTP). Implementing the MTP layer 2 (data link) functions in host software is difficult with multiple performance issues.

The SS7 controller is implemented using the MCC hardware with microcode running on the RISC. The MCC implements the following layer 2 portions of the MTP:

- Signal unit (SU) retransmission
- Automatic fill-in signal unit (FISU) transmission
- Short SU filtering
- Duplicate fill-in and link-status signal unit (FISU/LSSU) filtering
- Octet counting
- Signal unit error rate monitoring
- Good frame counter and bad frame counter
- Initial alignment (supports alignment error rate monitoring)

Additional host software is needed to handle the following higher-level functions of the MTP layer 2 not supported by the SS7 controller:

- Link state control
- Flow control

SS7 features are as follows:

- Up to 128 independent communication channels
- Independent mapping for receive and transmit
- Standard HDLC features
- Maintenance of signal unit error monitor (SUERM)
- Maintenance of alignment error rate monitor (AERM)
- Maintenance of separate counters for error-free and bad frames
- Detection and stripping of long signal units
- Discard of short signal units (less than 5 octets)
- Transmission of signal units with a programmable delay (applies to JT-Q.703 standard)
- Automatic transmission of fill-in signal units (FISU)
- Automatic retransmission of signal units (for link-status signal unit (LSSU) retransmission)
- Automatic discard of identical FISUs and LSSUs using a user-defined mask
- Octet counting mode in case of long signal units and receiver overrun
- Five circular interrupt tables with programmable size and overflow identification—one for transmit and four for receive.
- Global or individual channel loop modes
- Efficient bus usage (no bus usage for inactive channels or for active channels with nothing to send)
- Efficient control of interrupts to the QUICC Engine block
- Supports external BD tables
- Uses on-chip multi-port RAM for parameter storage
- Uses 64-bit data transactions for reading and writing data in BDs

34.1.3.4 AAL1 Circuit Emulation Service (CES)

When using AAL1 CES, the structured and unstructured data are transferred between the ATM and MCC automatically without QUICC Engine block intervention. This chapter will only describe the modifications required for the MCC. For the AAL1 CES configuration, see Chapter 35, "ATM AAL1 Circuit Emulation Service."

34.1.3.5 Superchannel

Superchannel is not a protocol dependant mode, rather a performance enhancing mode.

Superchanneling is used for sending multiple back-to-back timeslots from the same MCC channel. TX MCC SI entry is not recommended to be more than 2 bytes, due to the increase in required performance. So in every place that such frame structure is needed the superchannel scheme can be used. A single MCC channel is the combination of one MCC channel and one set of related channel parameters and buffer descriptors. A superchannel is the combination of multiple MCC TX channels and one set of MCC channel's parameters and buffer descriptors to manage this group of channels. This provides the ability to construct a larger overall channel than that of a single normal MCC TX channel. Superchannel makes use of HW resources of unused channels to enhance its performance, the user may NOT use the channels in the superchannel for any other purpose nor use the same channel twice in the construction of a superchannel.

The length of superchanneled timeslot may be one of the following sizes 5, 6, 7, 8 or 16 bits.

Note that in the same superchannel all the channels must be the same size.

34.2 Memory Map/Register Definition

The MCC uses the following data structures:

- Global MCC parameters (common to all 256 channels) placed in the MURAM from the offset (relative to the MURAM base address).
- Channel-specific parameters. Each channel uses 64 bytes of specific parameters placed in the MURAM at offset 64*CH_NUM (relative to the MURAM base address). In SS7 mode each channel uses 128 bytes of specific parameters placed in Multi-user RAM. However still the channel specific parameters are placed at offset 64*CH_NUM. Consequently only even channel number can be used with SS7.

  Channel-specific parameters are described in the following sections:

  — Section 34.2.2.1, "Channel-Specific HDLC Parameters"

  — Section 34.2.2.2, "Channel-Specific Transparent Parameters"

  — Section 34.2.2.4, "Channel-Specific SS7 Parameters"

  Note that the MURAM memory corresponding to the inactive channels can be used for other purposes.
- Channel extra parameters. Each channel uses 8 bytes of extra parameters placed in the MURAM at offset XTRABASE + 8*CH_NUM (relative to the MURAM base address). XTRABASE is one of the global MCC parameters.

Channel extra parameters as used in octet counting mode applies only to the ITU-T and ANSI standards. The SS7 microcode will not work if both the Japanese standard and OCM features are selected. Note that the MURAM memory corresponding to inactive channels can be used for other purposes.

- Superchannel table (used only if superchanneled timeslots are defined in SIRAM programming). This table is placed in the MURAM from the offset SCTPBASE (relative to the MURAM base address). SCTPBASE is one of the global MCC parameters. The superchannel table is described in Section 34.2.4, "Superchannel Table."
- BD tables placed in the external memory. All the BD tables associated with the MCC must reside in a 512-KByte segment. The absolute base addresses of a channel BD table is MCCBASE + 8*RBASE (for the receiver) and MCCBASE + 8*TBASE (for the transmitter). MCCBASE is one of the global MCC parameters and RBASE/TBASE are channel extra parameters. Each BD table is a circular queue. One BD includes status bits, start address and length of a data buffer. Figure 34-2 shows the BD structure for the MCC.
- Circular interrupt tables placed in external memory. There is one table for the transmitter interrupts (base address TINTBASE) and from one to four tables for receiver interrupts (base address RINTBASE0–RINTBASE3). TINTBASE and RINTBASE0–RINTBASE3 are global MCC parameters.
- Four registers (MCCE, MCCM, MERL, and MCCF)
  - Section 34.2.6.1, "MCC Event Register (MCCE)/Mask Register (MCCM)
  - Section 34.2.8, "MCC Emergency Request Level (MERL)
  - Section 34.2.5, "MCC Configuration Registers (MCCF)

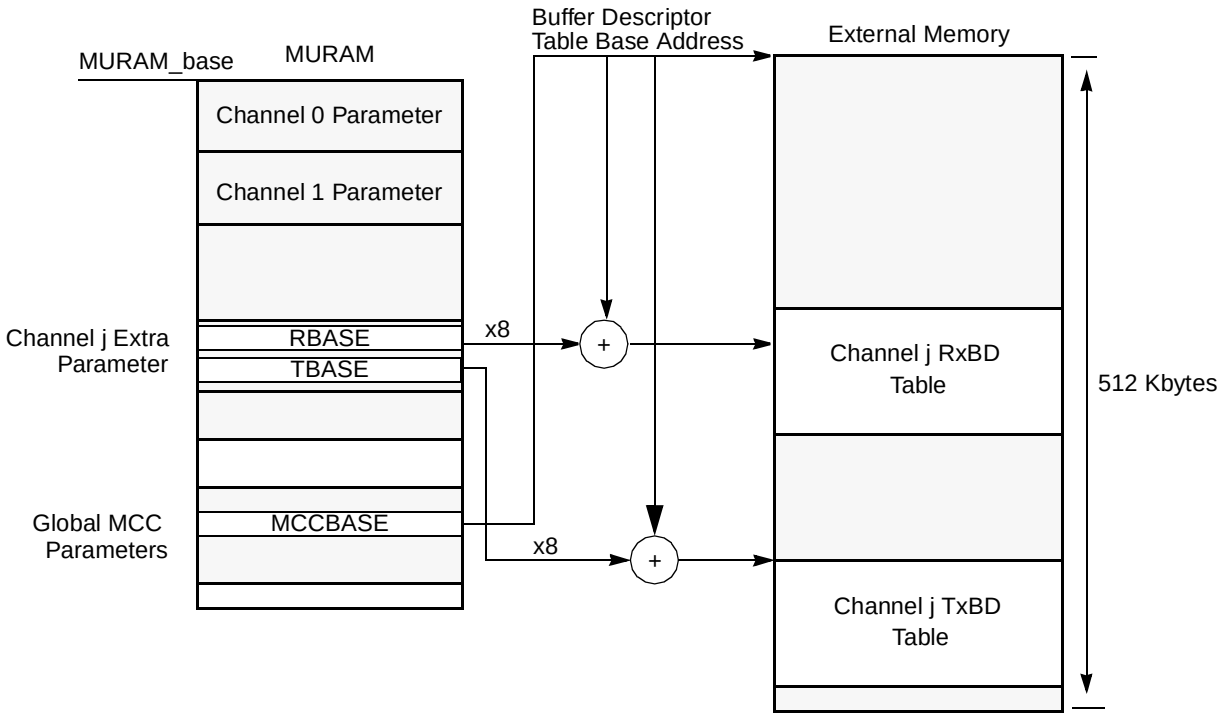


Figure 34-2. MCC BD Structure

Table 34-1. DMA Register Summary

Offset	Register	Access	Reset Value	Section/Page
General Registers				
0x00	MCC event register	R/W	0x0000_0000	34.2.6.1/34-38
0x04	MCC mask register	R/W	0x0000_0000	34.2.6.1/34-38
0x08	MCC configuration register	R/W	0x0000_0000	34.2.5/34-35
0x0C	MCC emergency request level	R/W	0x4000_4000	34.2.8/34-45

34.2.1 Global MCC Parameters

The global MCC parameters are described in Table 34-2.

Table 34-2. Global MCC Parameters

Offset[1]	Name	Width	Description
0x00	**MCCBASE**	Word	MCC base pointer. User-initialized parameter points to the starting address of a 512-Kbyte BD segment in external memory.
0x04	**MCCSTATE**	Hword	MCC state, used by the QUICC Engine block for global state definition. Should be cleared during initialization.
0x06	**MRBLR**	Hword	Maximum receive buffer length (user-initialized). 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 8.
0x08	**GRFTHR**	Hword	Global receive frame threshold. Used to reduce interrupt overhead that would otherwise occur when many short HDLC frames arrive, each causing an RXF interrupt. Programming GRFTHR can be used to limit the frequency of RXF interrupts. In normal operation, an unmasked RXF event is written to the interrupt table on each received frame. However, a user may choose to mask the RXF event and instead rely upon the RINTx event written to the interrupt table. This occurs only when the number of RXF events (by all channels) reaches the GRFTHR value. This parameter does not need to be reset after an interrupt.
0x0A	**GRFCNT**	Hword	Global receive frame count. A down counter used to implement the GRFTHR feature. It should be initialized to the GRFTHR value. The QUICC Engine block writes an entry in a circular interrupt table and decrements GRFCNT each time a frame is received. When GRFCNT reaches zero, the QUICC Engine block generates an interrupt and re-initializes GRFCNT with the GRFTHR value. This parameter does not need to be reset after an interrupt.
0x0C	RINTTMP	Word	Temporary location for holding the receive interrupt queue entry, used by the QUICC Engine block (reserved)
0x10	DATA0	Word	Temporary location for holding data, used by the QUICC Engine block (reserved)
0x14	DATA1	Word	Temporary location for holding data, used by the QUICC Engine block (reserved)
0x18	**TINTBASE**	Word	Multi-channel transmitter circular interrupt table base address. The interrupt circular table is a cyclic table (FIFO-like). Each table entry contains information about an interrupt request generated by the MCC to the host.
0x1C	**TINTPTR**	Word	Pointer to the transmitter circular interrupt table. The QUICC Engine block writes the next interrupt information to this entry when an exception occurs. The user must copy the TINTBASE value to TINTPTR before enabling interrupts. Further updates of the TINTPTR are done by the QUICC Engine block.

Table 34-2. Global MCC Parameters (continued)

Offset[1]	Name	Width	Description
0x20	**TINTTMP**	Word	Temporary location for holding the transmit interrupt queue entry, used by the QUICC Engine block. The system initializes this field before initializing the MCC. The user must clear it before enabling interrupts.
0x24	**SCTPBASE**	Hword	Internal pointer for the superchannel transmit table, offset from the MURAM address
0x26	**MCPBASE**	Hword	MCC Channel Pointer 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.
0x28	**C_MASK32**	Word	CRC constant (user initialized to 0xDEBB20E3). Used for 32-bit CRC-CCITT calculation if HDLC mode is chosen for a selected channel. (This option is programmable. For each HDLC channel, one of two CRC-CCITTs can be selected through the CHAMR, see Section 34.2.2.1.3, “Channel Mode Register (CHAMR)—HDLC Mode.”)
0x2C	**XTRABASE**	Hword	Pointer for the beginning of the extra parameters information, offset from the MURAM address
0x2E	**C_MASK16**	Hword	CRC constant (user initialized to 0xF0B8). Used for 16-bit CRC-CCITT calculation if HDLC mode is chosen for a selected channel. This option is programmable. For each HDLC channel, one of two CRC-CCITT can be selected through the CHAMR.
0x30	**RINTTMP0**	Word	RINTTMPx. Temporary location for holding a receive circular interrupt circular table entry (for tables 0–3), used by the QUICC Engine block. The user must clear it before enabling interrupts.
0x34	**RINTTMP1**	Word	
0x38	**RINTTMP2**	Word	
0x3C	**RINTTMP3**	Word	
0x40	**RINTBASE0**	Word	RINTBASEx—Multi-channel receiver circular interrupt table base address. The interrupt circular table is a cyclic table (FIFO-like). Each table entry contains information about an interrupt request generated by the MCC to the host. RINTPTRx—Pointer to the receiver circular interrupt table. The RISC writes the next interrupt information to this entry when an exception occurs. The user must copy the RINTBASEx value to RINTPTRx before enabling interrupts. Further updates of the RINTPTRx are done by the QUICC Engine block.
0x44	**RINTPTR0**	Word	
0x48	**RINTBASE1**	Word	
0x4C	**RINTPTR1**	Word	
0x50	**RINTBASE2**	Word	
0x54	**RINTPTR2**	Word	
0x58	**RINTBASE3**	Word	
0x5C	**RINTPTR3**	Word	
0x60	TS_TMP	Word	Temporary place for time stamp

1 Offset to MCC Base

34.2.2 Channel-Specific Parameters

Each MCC ROUTER in the MCC is managed by a set of channel-specific parameters. These parameters can change based upon what protocol is being used on that channel. The following sections describe the various programming models used for an MCC channel.

34.2.2.1 Channel-Specific HDLC Parameters

Table 34-3 describes channel-specific parameters for HDLC.

Table 34-3. Channel-Specific Parameters for HDLC

Offset[1]	Name	Width	Description
0x00	**TSTATE**	Word	Tx internal state. To start a transmitter channel the user must write to TSTATE 0x*HH*80_0000. HH is the TSTATE high byte described in Section 34.2.2.1.1, "Internal Transmitter State (TSTATE)—HDLC Mode"
0x04	**ZISTATE**	Word	Zero-insertion machine state. User-initialized to one of the following values: 0x10000207 for regular channel transmitting all 1s before first frame of data 0x00000207 for regular channel transmitting flags before first frame of data 0x30000207 for inverted channel transmitting all 1s before first frame of data 0x20000207 for inverted channel transmitting flags before first frame of data **Note:** Used in conjunction with ZIDATA0 and ZIDATA1.
0x08	**ZIDATA0**	Word	Zero-insertion high word data buffer. User-initialized to one of the following values: 0xFFFFFFFF allows transmission of all 1s before first frame of data 0x7E7E7E7E allows transmission of flags before first frame of data **Note:** Used in conjunction with ZISTATE and ZIDATA1.
0x0C	**ZIDATA1**	Word	Zero-insertion low word data buffer. User-initialized to one of the following values: 0xFFFFFFFF allows transmission of all 1s before first frame of data 0x7E7E7E7E allows transmission of flags before first frame of data **Note:** Used in conjunction with ZISTATE and ZIDATA0.
0x10	TBDFlags	Hword	TxDB flags, used by the QUICC Engine block (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to the current absolute data address of channel, used by the QUICC Engine block (read-only for the user)
0x18	**INTMSK**	Hword	Channel's interrupt mask flag. See Section 34.2.2.3.3, "Interrupt Circular Table Entry and Interrupt Mask (INTMSK)—AAL1 CES"
0x1A	**CHAMR**	Hword	Channel mode register. See Section 34.2.2.1.3, "Channel Mode Register (CHAMR)—HDLC Mode"
0x1C	TCRC	Word	Temp transmit CRC. Temp value of CRC calculation result, used by the QUICC Engine block (read-only for the user)
0x20	**RSTATE**	Word	Rx internal state. To start a receiver channel the user must write to RSTATE 0x*HH*80_0000. *HH* is the value of the RSTATE high byte described in Section 34.2.2.1.4, "Internal Receiver State (RSTATE)—HDLC Mode"
0x24	**ZDSTATE**	Word	Zero-deletion machine state (User-initialized to 0x00FFFFE0 for regular channel and 0x20FFFFE0 for inverted channel)
0x28	**ZDDATA0**	Word	Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)
0x2C	**ZDDATA1**	Word	Zero-deletion low word data buffer (User-initialized to 0x8000FFFF)
0x30	RBDFlags	Hword	RxBD flags, used by the QUICC Engine block (read-only for the user)
0x32	RBDCNT	Hword	Rx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)

Table 34-3. Channel-Specific Parameters for HDLC (continued)

Offset[1]	Name	Width	Description
0x34	RBDPTR	Word	Rx internal data pointer. Points to current absolute data address of channel, used by the QUICC Engine block (read-only for the user)
0x38	**MFLR**	Hword	Maximum frame length register. Defines the longest expectable frame for this channel. (64-Kbyte maximum). The remainder of a frame that is larger than MFLR is discarded and the LG flag is set in the last frame's BD. An interrupt request might be generated (RXF and RXB) depending on the interrupt mask. A frame's length is considered to be everything between flags, including the CRC. No more data is written into the current buffer when the MFLR violation is detected.
0x3A	MAX_CNT	Hword	Max_length counter, used by the QUICC Engine block (read-only for the user)
0x3C	RCRC	Word	Temp receive CRC, used by the QUICC Engine block (read-only for the user)

[1] The offset is relative to Multi-User RAM (MURAM) base address +MCPBASE+ 64*CH_NUM

34.2.2.1.1 Internal Transmitter State (TSTATE)—HDLC Mode

The internal transmitter state (TSTATE) is a 4-byte register that provides the transaction parameters associated with SDMA channel accesses (like the bus mode registers). It also starts the transmitter channel.

To start the channel, write 0x*HH*800000 to TSTATE, where *HH* is the TSTATE high byte (see Figure 34-3). When the channel is active, the QUICC Engine block changes the value of the three LSBs, hence these 3 bytes must be masked if the user reads back the TSTATE.

Access: User read-write

	0	1	2	3	4	5	6	7
R W	—t		**GBL**	**BO**		**CETM**	**DTB**	**BDB**
Reset:	0	0	0	0	0	0	0	0

Figure 34-3. TSTATE High Byte

TSTATE high byte fields are described in Table 3-6.

Table 34-4. TSTATE High-Byte Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global. Setting GBL activates snooping (only the coherent system bus can be snooped, this parameter is ignored for secondary bus transactions). To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "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 or at the beginning of the next BD. 00, 01, 11 Reserved 10 Big-endian

Table 34-4. TSTATE High-Byte 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 block, for debug purposes.
6	**DTB**	Data bus indicator. Selects the bus that handles transfers to and from data buffers. 0 Coherent system bus SDMA 1 QUICC Engine block secondary bus SDMA
7	**BDB**	BD bus. Selects the bus that handles transfers to/from BD and interrupt circular tables. 0 Coherent system bus SDMA used for accessing BDs 1 QUICC Engine block secondary bus SDMA used for accessing BDs

34.2.2.1.2 Interrupt Mask (INTMSK)—HDLC Mode

The interrupt mask (INTMSK) provides bits for enabling/disabling the reporting of each possible event defined in the interrupt circular table entry. For descriptions of each event bit, refer to Section 34.2.6.2, "Interrupt Circular Table Entry."

	0–5	6	7	8	9	10	11	12	13	14	15
Interrupt Entry	—	UN	TXB	—		NID	IDL	MRF	RXF	BSY	RXB
INTMSK	—	**Mask Bits**		—		**Mask Bits**					

Figure 34-4. INTMSK Mask Bits

To enable an interrupt, set the corresponding bit. If a bit is cleared, no interrupt request is generated and no new entry is written in the circular interrupt table. The user must initialize INTMSK prior to operation. Reserved bits should remain cleared.

34.2.2.1.3 Channel Mode Register (CHAMR)—HDLC Mode

The channel mode register (CHAMR) is a user-initialized register, shown in Figure 3-5. For a descriptions of CHAMR in transparent and SS7 modes, refer to Section 3.2.2.3 and Section 3.2.4.1 respectively. For channels that are used in conjunction with CES functionality, the user should refer to Section 34.2.2.3.4, "Channel Mode Register (CHAMR)—AAL1 CES," for additional information.

Offset: 0x1A — Access: User read-write

	0	1	2	3	4	5	6–7	8	9	10	11	12–13	14–15
R W	**MODE**	**POL**	**1**	**IDLM**	—	**RD**	—	**CRC**	—	**TS**	**RQN**	**NOF**	
Reset	0	0	0	0	0	0	0 0	0	0	0	0	0 0	0 0

Figure 34-5. Channel Mode Register (CHAMR)—HDLC Mode

CHAMR fields are described in Table 34-5.

Table 34-5. CHAMR Field Descriptions

Bits	Name	Description
0	**MODE**	This mode bit determines whether the HDLC or transparent mode is used. It also determines how other CHAMR bits are interpreted. 0 Transparent mode. See Section 34.2.2.2.3, "Channel Mode Register (CHAMR)—Transparent Mode" 1 HDLC mode. See Section 34.2.2.1.3, "Channel Mode Register (CHAMR)—HDLC Mode"
1	**POL**	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The QUICC Engine block does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL is used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The QUICC Engine block clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To minimize useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
2	**1**	Must be set.
3	**IDLM**	Idle mode. 0 No idle patterns are sent between frames. After sending NOF+1 flags, the transmitter starts sending the data of the frame. If the transmission is between frames and the frame buffers are not ready, the transmitter sends flags until it can start transmitting the data. 1 At least one idle pattern is sent between adjacent frames. The NOF value shall be no smaller than the PAD setting, see Section 34.2.7.2, "Transmit Buffer Descriptor (TxBD)." If NOF = 0, this is identical to flag sharing in HDLC. Mode flags precede the actual data. When IDLM = 1, at least one idle pattern is sent between adjacent frames. If the transmission is 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 IDLE mode is selected and NOF = 1, the following sequence is sent: .....init value, FF flag data,..... The init value before the idle will be ones.
4	—	This bit must be cleared.
5	**RD**	0 Normal bit order (transmit/receive the lsb of each octet first) 1 Reversed bit order (transmit/receive the msb of each octet first)
6–7	—	These bits must be cleared.
8	**CRC**	Selects the type of CRC when HDLC channel mode is used. 0 16-bit CCITT-CRC 1 32-bit CCITT-CRC
9	—	This bit must be cleared.
10	**TS**	Receive time stamp. If this bit is set, a 4 byte time stamp is written at the beginning of every data buffer pointed to by the BD. Also, the data buffer must start from an address equal to 8*n-4 (n is any integer larger than 0).

Table 34-5. CHAMR Field Descriptions (continued)

Bits	Name	Description
11–12	**RQN**	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0. 01 Queue number 1. 10 Queue number 2. 11 Queue number 3.
13–15	**NOF**	Number of flags. NOF defines the minimum number of flags before frames: 000 At least 1 flag 001 At least 2 flags .... 111 At least 8 flags

34.2.2.1.4 Internal Receiver State (RSTATE)—HDLC Mode

Internal receiver state (RSTATE) is a 4-byte register that provides transaction parameters associated with SDMA channel accesses (like the bus mode registers) and starts the receiver channel.

To start the channel the user must write 0x*HH*800000 to RSTATE, where *HH* is the RSTATE high byte (see Figure 3-6). When the channel is active, the QUICC Engine block changes the value of the 3 LSBs, hence these 3 bytes must be masked if the user reads back the RSTATE.

Offset: 0x20 Access: User read-write

	0	1	2	3	4	5	6	7
R W	—		**GBL**	**BO**		**CETM**	**DTB**	**BDB**
Reset:	0	0	0	0	0	0	0	0

Figure 34-6. Rx Internal State (RSTATE) High Byte

RSTATE high byte fields are described in Table 34-6.

Table 34-6. RSTATE High Byte Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global. Setting GLB activates snooping (only the coherent system bus can be snooped, this parameter is ignored for secondary bus transactions). To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, “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 or at the beginning of the next BD. 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.

Table 34-6. RSTATE High Byte Field Descriptions (continued)

Bits	Name	Description
6	**DTB**	Data bus indicator. The transfers to data buffers are handled by the: 0 Coherent system bus SDMA 1 QUICC Engine block secondary bus SDMA
7	**BDB**	BD and interrupt circular tables bus indicator. The transfers to/from BD and interrupt circular tables are handled by the: 0 Coherent system bus SDMA 1 QUICC Engine block secondary bus SDMA **Note:** The following restrictions result from the fact that there is a common bus selection bit for BDs and interrupt circular tables: • The RxBDs of all the channels that use a particular interrupt table must reside on the same bus (memory or secondary). • All TxBDs must reside on the same bus (memory or secondary).

34.2.2.2 Channel-Specific Transparent Parameters

Table 34-7 describes channel-specific parameters for transparent operation.

Table 34-7. Channel-Specific Parameters for Transparent Operation

Offset[1]	Name	Width	Description
0x00	**TSTATE**	Word	Tx internal state. To start a transmitter channel the user must write to TSTATE 0x*HH*80_0000. *HH* is the TSTATE high byte described in Section 34.2.2.1.1, "Internal Transmitter State (TSTATE)—HDLC Mode"
0x04	**ZISTATE**	Word	Zero-insertion machine state. (User-initialized to 0x10000207 for regular channel, or 0x30000207 for inverted channel)
0x08	**ZIDATA0**	Word	Zero-insertion high word data buffer (User-initialized to 0xFFFFFFFF)
0x0C	**ZIDATA1**	Word	Zero-insertion low word data buffer (User-initialized to 0xFFFFFFFF)
0x10	TBDFlags	Hword	TxDB flags, used by the QUICC Engine block (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to current absolute data address of channel, used by the QUICC Engine block (read-only for the user)
0x18	**INTMSK**	Hword	Channel's interrupt mask flag. See Section 34.2.2.2.2, "Interrupt Mask (INTMSK)—Transparent Mode"
0x1A	**CHAMR**	Hword	Channel mode register. See Section 34.2.2.2.3, "Channel Mode Register (CHAMR)—Transparent Mode"
0x1C	—	Word	Reserved
0x20	**RSTATE**	Word	Rx internal state. To start a receiver channel the user must write to RSTATE 0x*HH*80_0000. *HH* is the RSTATE high byte described in Section 34.2.2.1.4, "Internal Receiver State (RSTATE)—HDLC Mode"

Table 34-7. Channel-Specific Parameters for Transparent Operation (continued)

<table>
<tr><th>Offset[1]</th><th>Name</th><th>Width</th><th>Description</th></tr>
<tr><td>0x24</td><td>ZDSTATE</td><td>Word</td><td>Zero-deletion machine state. Initialize ZDSTATE as in the following table:
<table>
<tr><th>Channel Type</th><th>RCVSYNC</th><th>ZDSTATE Initial Programmed Value</th></tr>
<tr><td colspan="3">If pattern synchronization is used (CHAMR[SYNC] = 1x), then…</td></tr>
<tr><td rowspan="2">Regular</td><td>0bxxxx_xxxx_xxxx_xxx0</td><td>0x00FF_FFE0</td></tr>
<tr><td>0bxxxx_xxxx_xxxx_xxx1</td><td>0x0000_0000</td></tr>
<tr><td rowspan="2">Inverted</td><td>0bxxxx_xxxx_xxxx_xxx0</td><td>0x20FF_FFE0</td></tr>
<tr><td>0bxxxx_xxxx_xxxx_xxx1</td><td>0x2000_0000</td></tr>
<tr><td colspan="3">If pattern synchronization is not used (CHAMR[SYNC] = 00), then…</td></tr>
<tr><td>Regular</td><td>0x0000</td><td>0x00FF_FFE0</td></tr>
<tr><td>Inverted</td><td>0x0000</td><td>0x20FF_FFE0</td></tr>
<tr><td colspan="3">If slot synchronization is used (CHAMR[SYNC] = 01), then…</td></tr>
<tr><td>Regular</td><td>0x0000</td><td>0x002F_DFE0</td></tr>
<tr><td>Inverted</td><td>0x0000</td><td>0x202F_DFE0</td></tr>
</table></td></tr>
<tr><td>0x28</td><td>ZDDATA0</td><td>Word</td><td>Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)</td></tr>
<tr><td>0x2C</td><td>ZDDATA1</td><td>Word</td><td>Zero-deletion low word data buffer (User-initialized to 0x8000FFFF)</td></tr>
<tr><td>0x30</td><td>RBDFlags</td><td>Hword</td><td>RxBD flags, used by the QUICC Engine block (read-only for the user)</td></tr>
<tr><td>0x32</td><td>RBDCNT</td><td>Hword</td><td>Rx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)</td></tr>
<tr><td>0x34</td><td>RBDPTR</td><td>Word</td><td>Rx internal data pointer. Points to current absolute data address of channel, used by the QUICC Engine block (read-only for the user)</td></tr>
<tr><td>0x38</td><td>TMRBLR</td><td>Hword</td><td>Transparent maximum receive buffer length. Defines the maximum number of bytes written to a receiver buffer before moving to the next buffer for the respective channel. This value must be 8 byte aligned.</td></tr>
<tr><td>0x3A</td><td>RCVSYNC</td><td>Hword</td><td>Receive synchronization pattern. Defines the synchronization pattern when CHAMR[SYNC] is 0b1x. The two bytes are checked in reverse order (byte from address 0x3B first and byte from address 0x3A last). Non-inverted data is used for synchronization even if the channel is programmed to invert the data. When CHAMR[SYNC] = 0b01 (slot synchronization) program RCVSYNC to 0xFF7E. Clear RCVSYNC when CHAMR[SYNC] = 0b00.</td></tr>
<tr><td>0x3C</td><td>—</td><td>Word</td><td>Reserved</td></tr>
</table>

[1] The offset is relative to Multi-user RAM address +MCPBASE+64*CH_NUM

34.2.2.2.1 Internal Transmitter State (TSTATE)—Transparent Mode

In transparent mode, TSTATE functions the same as in HDLC mode. For a description, refer to Section 34.2.2.1.1, “Internal Transmitter State (TSTATE)—HDLC Mode.”

34.2.2.2.2 Interrupt Mask (INTMSK)—Transparent Mode

In transparent mode, INTMSK functions the same as in HDLC mode. For a description, refer to Section 34.2.2.1.2, “Interrupt Mask (INTMSK)—HDLC Mode.”

34.2.2.2.3 Channel Mode Register (CHAMR)—Transparent Mode

Figure 3-7 shows the user-initialized channel mode register, CHAMR, for transparent mode. For channels that are used in conjunction with CES functionality, the user should refer to Section 34.2.2.3.4, “Channel Mode Register (CHAMR)—AAL1 CES,” for additional information.

Offset: 0x1A Access: User read-write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MODE	POL	1	1	EP	RD	SYNC		—		TS	RQN		—		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-7. Channel Mode Register (CHAMR)—Transparent Mode

CHAMR fields are described in Table 34-8.

Table 34-8. CHAMR Field Descriptions—Transparent Mode

Bits	Name	Description
0	**MODE**	Channel mode. Selects either HDLC or transparent mode. 0 Transparent mode. 1 HDLC mode
1	**POL**	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The QUICC Engine block does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL is used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The QUICC Engine block clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
2–3	**0b11**	Must be set.
4	**EP**	Empty polarity and enable polling. 0 The E bit in the RxBD is handled in positive logic (1 = empty; 0 = not empty). Polling occurs only if POL is set. 1 The E bit in the RxBD is handled in negative logic (0 = empty, 1 = not empty). Polling occurs disregarding the value of POL.
5	**RD**	0 Normal bit order (transmit/receive the lsb of each octet first) 1 Reversed bit order (transmit/receive the msb of each octet first)

Table 34-8. CHAMR Field Descriptions—Transparent Mode (continued)

Bits	Name	Description			
6–7	**SYNC**	Synchronization. SYNC controls synchronization of multi-channel operation in transparent mode.			
		SYNC	**Receive**	**Transmit**	**Description**
		00	None	None	Transmitter and receiver operate with no synchronization algorithm. RCV SYNC should be cleared or received data may be shifted.
		01	Slot	Slot	The first data is sent/received in the slot defined in the slot assignment table (for superchannels only). RCV SYNC should be cleared or received data may be shifted.
		10	8-bit	None	Receive data synchronization uses an 8-bit pattern specified by the 8 msb of RCV SYNC. The sync bytes are not written to the receive buffer.
		11	16-bit	None	Receive data synchronization uses a 16-bit pattern specified by RCV SYNC. The first byte of the sync pattern will not be written to the receive buffer. The second byte of the sync pattern will be written to the receive buffer (first and second represent the order in which the two bytes of the sync pattern are received on the serial channel).
8–9	—	Reserved, must be cleared.			
10	**TS**	Receive time stamp. If this bit is set, a 4 byte time stamp is written at the beginning of every data buffer that the BD points to. Also, the data buffer must start from an address equal to 8*N-4 (N is any number larger than 0).			
11–12	**RQN**	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0. 01 Queue number 1. 10 Queue number 2. 11 Queue number 3.			
13–15	—	Reserved, must be cleared.			

34.2.2.2.4 Internal Receiver State (RSTATE)—Transparent Mode

In transparent mode, RSTATE functions the same as in HDLC mode. For a description, refer to Section 34.2.2.1.4, “Internal Receiver State (RSTATE)—HDLC Mode.”

34.2.2.3 MCC Parameters for AAL1 CES Usage

When using AAL1 circuit emulation service (CES), the structured and unstructured data are transferred between the ATM and MCC automatically without QUICC Engine block intervention. Refer to Chapter 35, “ATM AAL1 Circuit Emulation Service.” The following subsections describe the additional parameters required for AAL1 CES.

34.2.2.3.1 MCC Global Parameters—AAL1 CES

The global MCC parameters specific to AAL1 CES operation are described in Table 34-9.

Table 34-9. CES-Specific Global MCC Parameters

Offset[1]	Name	Width	Description
0x00	**CATB**	Hword	CES adaptive threshold tables base address. Points to the multi-user RAM area containing the CES slip control thresholds and the adaptive counterl, see Section 35.5, "ATM-to-TDM Adaptive Slip Control." Should be 8-byte aligned (8 octets for each AAL1-MCC channel). User-defined and should match the CATB value programmed in the AAL1 parameter RAM; see Section 35.8.1, "AAL1 CES Parameter RAM."
0x02	—	Hword	Reserved, should be cleared during initialization.
0x04, 0x08, 0x0C, 0x10, 0x14, 0x18, 0x1C, 0x20	**UTA1, UTA2, UTA3, UTA4, UTA5, UTA6, UTA7, UTA8**	Hword	Underrun template address for Group*X*. Points to the multi-user RAM area containing the user-defined template to be sent during an MCC transmitter pre-underrun condition.
0x06, 0x0A, 0x0E, 0x12, 0x16, 0x1A, 0x1E, 0x22,	**UTS1, UTS2, UTS3, UTS4, UTS5, UTS6, UTS7. UTS8**	Hword	Underrun template size for Group*X*. This is the size in bytes of the underrun template buffer.

[1] The offset to the CES-specific global MCC parameter RAM is +0x0080 from regular offset.

34.2.2.3.2 Channel-Specific Parameters—AAL1 CES

The following are changes that occur in the channel-specific parameter RAM when using AAL1 CES.

34.2.2.3.3 Interrupt Circular Table Entry and Interrupt Mask (INTMSK)—AAL1 CES

Interrupt circular table entries contain information about channel-specific events. The interrupt mask (INTMSK) provides bits for enabling/disabling the reporting of each possible event defined in the interrupt circular table entry. Note that two CES-related interrupts provide slip indications for the MCC transmitter. They are reflected in both the interrupt circular table entries and the INTMSK fields, as described in Table 34-19. Figure 34-8 shows the INTMSK mask bits.

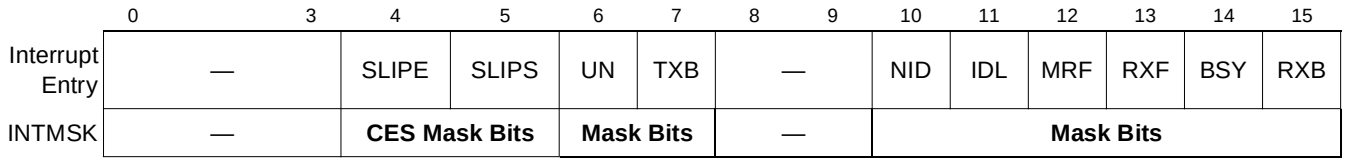


Figure 34-8. INTMSK Mask Bits

To enable an interrupt, set the corresponding bit. If a bit is cleared, no interrupt request is generated and no new entry is written in the interrupt circular table. The user must initialize INTMSK prior to operation. Reserved bits are cleared.

34.2.2.3.4 Channel Mode Register (CHAMR)—AAL1 CES

Figure 3-9 shows the user-initialized channel mode register, CHAMR, for CES operation. It is the same as the CHAMR in transparent mode with three extra CES fields in bits 13–15.

Offset: 0x1A Access: User read-write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MODE	POL	1	1	EP	RD	SYNC		—		TS	RQN		CESM	UDC	UTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-9. Channel Mode Register (CHAMR)—AAL1 CES

The CHAMR in CES mode fields are described in Table 34-10.

Table 34-10. CHAMR Field Descriptions—CES Mode

Bits	Name	Description
0	**MODE**	Channel mode. Selects either HDLC or transparent mode. Must be cleared for CES operation. 0 Transparent mode. 1 HDLC mode
1	**POL**	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The QUICC Engine block does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL is used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The QUICC Engine block clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
2–3	**0b11**	Must be set.
4	**EP**	Empty polarity and enable polling. 0 The E bit in the RxBD is handled in positive logic (1 = empty; 0 = not empty). Polling occurs only if POL is set. 1 The E bit in the RxBD is handled in negative logic (0 = empty, 1 = not empty). Polling occurs disregarding the value of POL.
5	**RD**	0 Normal bit order (transmit/receive the lsb of each octet first) 1 Reversed bit order to be reversed (transmit/receive the msb of each octet first).

Table 34-10. CHAMR Field Descriptions—CES Mode (continued)

<table>
<tr><th>Bits</th><th>Name</th><th colspan="4">Description</th></tr>
<tr><td>6–7</td><td>SYNC</td><td colspan="4">Synchronization. SYNC controls synchronization of multi-channel operation in transparent mode.</td></tr>
<tr><td></td><td></td><td>SYNC</td><td>Receive</td><td>Transmit</td><td>Description</td></tr>
<tr><td></td><td></td><td>00</td><td>None</td><td>None</td><td>Transmitter and receiver operate with no synchronization algorithm</td></tr>
<tr><td></td><td></td><td>01</td><td>Slot</td><td>Slot</td><td>The first data is sent/received in the slot defined in the slot assignment table (for superchannels only)</td></tr>
<tr><td></td><td></td><td>10</td><td>8-bit</td><td>None</td><td>Receive data synchronization uses an 8-bit pattern specified by the 8 msb of RCVSYNC. The sync bytes will not be written to the receive buffer</td></tr>
<tr><td></td><td></td><td>11</td><td>16-bit</td><td>None</td><td>Receive data synchronization uses a 16-bit pattern specified by RCVSYNC. The first byte of the sync pattern will not be written to the receive buffer. The second byte of the sync pattern will be written to the receive buffer (first and second represent the order in which the two bytes of the sync pattern are received on the serial channel).</td></tr>
<tr><td>8–9</td><td>—</td><td colspan="4">Reserved, should be cleared during initialization.</td></tr>
<tr><td>10</td><td>TS</td><td colspan="4">Receive time stamp. If this bit is set a 4 byte time stamp is written at the beginning of every data buffer that the BD points to.If this bit is set the data buffer must start from an address equal to 8*N-4 (N is any number larger than 0).</td></tr>
<tr><td>11–12</td><td>RQN</td><td colspan="4">Receive queue number. Specifies the receive interrupt queue number.
00 Queue number 0
01 Queue number 1
10 Queue number 2
11 Queue number 3</td></tr>
<tr><td>13</td><td>CESM</td><td colspan="4">Circuit emulation service mode.
0 Normal mode
1 CES mode</td></tr>
<tr><td>14</td><td>UDC</td><td colspan="4">User-defined cell support.
0 User-defined ATM cells are not supported.
1 User-defined ATM cells are supported.</td></tr>
<tr><td>15</td><td>UTM</td><td colspan="4">Underrun template mode.
0 Retransmit the last buffer.
1 Send the user-defined template.</td></tr>
</table>

34.2.2.4 Channel-Specific SS7 Parameters

Table 34-11 describes channel-specific parameters for SS7. Note that a given parameter location may have a different definition depending on the standard used (ITU-T/ANSI or Japanese standard).

Table 34-11. Channel-Specific Parameters for SS7

Offset[1]	Name	Width	Description
0x00	TSTATE	Word	Tx internal state. The user must write to TSTATE 0x*HH*80_0000. *HH* is the TSTATE High Byte. Refer to Section 34.2.2.1.1, "Internal Transmitter State (TSTATE)—HDLC Mode"
0x04	ZISTATE	Word	Zero-insertion machine state. User-initialized to one of the following values: 0x10000207 for regular channel transmitting all 1s before first frame of data 0x00000207 for regular channel transmitting flags before first frame of data 0x30000207 for inverted channel transmitting all 1s before first frame of data 0x20000207 for inverted channel transmitting flags before first frame of data **Note:** Used in conjunction with ZIDATA0 and ZIDATA1.
0x08	ZIDATA0	Word	Zero-insertion high word data buffer. User-initialized to one of the following values: 0xFFFFFFFF allows transmission of all 1s before first frame of data 0x7E7E7E7E allows transmission of flags before first frame of data **Note:** Used in conjunction with ZISTATE and ZIDATA1.
0x0C	ZIDATA1	Word	Zero-insertion low word data buffer. User-initialized to one of the following values: 0xFFFFFFFF allows transmission of all 1s before first frame of data 0x7E7E7E7E allows transmission of flags before first frame of data **Note:** Used in conjunction with ZISTATE and ZIDATA0.
0x10	TBDFlags	Hword	TxBD flags, used by the QUICC Engine block (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to current absolute data address of channel, used by the QUICC Engine block (read-only for the user)
0x18	ECHAMR	Word	Extended channel mode register. See Section 34.2.2.4.1, "Extended Channel Mode Register (ECHAMR)—SS7 Mode"
0x1C	TCRC	Word	Temporary transmit CRC. Temporary value of CRC calculation result, used by the QUICC Engine block (read-only for the user)
0x20	RSTATE	Word	Rx internal state. To start a receiver channel the user must write to RSTATE 0x*HH*80_0000. *HH* is the RSTATE High Byte. Refer to Section 34.2.2.1.4, "Internal Receiver State (RSTATE)—HDLC Mode"
0x24	ZDSTATE	Word	Zero-deletion machine state (User-initialized to 0x00FFFFE0 for regular channel, or 0x20FFFFE0 for reversed bit order channel)
0x28	ZDDATA0	Word	Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)
0x2C	ZDDATA1	Word	Zero-deletion low word data buffer (User-initialized to 0x8000FFFF)
0x30	RBDFlags	Hword	RxBD flags, used by the QUICC Engine block (read-only for the user)
0x32	RBDCNT	Hword	Rx internal byte count. Number of remaining bytes in buffer, used by the QUICC Engine block (read-only for the user)
0x34	RBDPTR	Word	Rx internal data pointer. Points to current absolute data address of channel, used by the QUICC Engine block (read-only for the user)

Table 34-11. Channel-Specific Parameters for SS7 (continued)

Offset[1]	Name	Width	Description
0x38	MFLR	Hword	Maximum frame length register. Defines the longest expected frame for this channel. (64-Kbyte maximum). The remainder of a frame that is larger than MFLR is discarded and the LG flag is set in the last frame's BD. An interrupt request might be generated (RXF and RXB) depending on the interrupt mask. A frame's length is considered to be everything between flags, including CRC. No more data is written into the current buffer when the MFLR violation is detected.
0x3A	MAX_cnt	Hword	Max_length counter, used by the QUICC Engine block (read-only for the user)
0x3C	RCRC	Word	Temporary receive CRC, used by the QUICC Engine block (read-only for the user)
0x40	N	Hword	Applies to ITU-T/ANSI SS7 only. Interrupt threshold in octet counting mode (N=16). See Section 34.2.2.4.2, "Signal Unit Error Monitor (SUERM)—SS7 Mode"
	N_cnt	Hword	Applies to ITU-T/ANSI SS7 only. Temporary down counter for N (user initialized to the value of N).
	JTSTTmp	Word	Applies to Japanese SS7 only. Temporary storage for Time-Stamp Register Value. Used by the QUICC Engine block to implement a 24-ms delay before sending FISU.
0x44	D	Hword	Signal unit to signal unit error ratio (SUERM parameter, user initialized to 256). See Section 34.2.2.4.2, "Signal Unit Error Monitor (SUERM)—SS7 Mode"
0x46	D_cnt	Hword	Applies to ITU-T/ANSI SS7 only. Temporary down-counter for D (user initialized to the value of D). D_cnt is decremented only when receive buffers are available.
	JTTDelay		Applies to Japanese SS7 only. FISU retransmission delay (specified in units of 512¨µs). According to the Japanese SS7 standard, the delay should be 24 ms and thus JTTDelay should be programmed to 24 ms/512 µs = 46.875 (approximately 47). Therefore, the user should program JTTDelay to 0x2F and the CETSCR to generate a 1 µs time stamp period. Refer to "QUICC Engine Time-Stamp Control Register (CETSCR)".
0x48	Mask1	Word	Mask for SU filtering, bytes 0-3. See Section 34.2.2.4.8, "SU Filtering—SS7 Mode"
0x4C	Mask2	Hword	Mask for SU filtering, byte 4. See Section 34.2.2.4.8, "SU Filtering—SS7 Mode"
0x4E	SS7_OPT	Hword	SS7 configuration register. See Section 34.2.2.4.4, "SS7 Configuration Register—SS7 Mode"
0x50	LRB1_Tmp	Word	Temporary storage, used by QUICC Engine block for SU filtering.
0x54	LRB2_Tmp	Hword	Temporary storage, used by QUICC Engine block for SU filtering.
0x56	SUERM	Hword	Signal unit error rate monitor counter (user initialized to 0). See Section 34.2.2.4.2, "Signal Unit Error Monitor (SUERM)—SS7 Mode"
0x58	LRB1	Word	Four first bytes of last received signal unit. Used by CP for SU filtering. See Section 34.2.2.4.8, "SU Filtering—SS7 Mode."
0x5C	LRB2	Hword	Fifth byte of last received signal unit. Used by QUICC Engine block for SU filtering. See Section 34.2.2.4.8, "SU Filtering—SS7 Mode"
0x5E	T	Hword	SUERM threshold value (user initialized to 64). See Section 34.2.2.4.2, "Signal Unit Error Monitor (SUERM)—SS7 Mode"
0x60	LHDR	Word	The BSN, BIB, FSN, FIB fields of last transmitted signal unit and result of CRC. Used by QUICC Engine block for automatic FISU transmission.
0x64	LHDR_Tmp	Word	Temporary storage, used by QUICC Engine block for automatic FISU transmission.

Table 34-11. Channel-Specific Parameters for SS7 (continued)

Offset[1]	Name	Width	Description
0x68	EFSUC	Word	Error-free signal unit counter, user initialized to 0. The counter is incremented whenever an error-free (no CRC error, no non-octet aligned error, no short or long frame errors) signal unit is received.
0x6C	SUEC	Word	Signal unit error counter, user initialized to 0. Incremented each time an SU is received that contains an error. These errors are: short frame, long frame, CRC error, and non-octet aligned error.
0x70	SS7STATE	Word	Internal state of SS7 controller, user initialized to 0.
0x74	JTSRTmp	Word	Temporary storage for time-stamp register value. Applies to Japanese SS7 only; otherwise should be cleared. Used by the CP to implement the 24-ms delay for signal unit error rate monitoring in Japanese SS7.
0x78	JTRDelay	Hword	FISU transmit delay (specified in units of 512us). Applies to Japanese SS7 only; otherwise should be cleared. According to the Japanese SS7 standard, the delay should be 24 ms and thus JTRDelay should be programmed to 24 ms/512 µs = 46.875 (approximately 47). Therefore the user should program JTRDelay to 0x2F and the CETSCR to generate a 1 µs time stamp period. Refer to “QUICC Engine Time-Stamp Control Register (CETSCR)”.
0x7A	M	Hword	ITU threshold for AERM. If M_cnt reaches M, an AERM interrupt is generated. Note that M is normally programmed to 5.
0x7C	M_cnt	Hword	Up-counter for M. Should be cleared during initialization.

1 The offset is relative to the multi-port RAM address +MCPBASE+ 64*CH_NUM. SS7 channel specific parameters require twice the amount of multi-port RAM required for HDLC or Transparent channel specific parameters. Therefore for SS7 even channel numbers (0, 2, 4, etc.) must be used and odd channel number must be left unused.Items in **boldface** must be initialized by the user. Unless otherwise stated, all other items are managed by microcode and should be initialized to zero.

34.2.2.4.1 Extended Channel Mode Register (ECHAMR)—SS7 Mode

The extended channel mode register (ECHAMR) is a user-initialized register, shown in Figure 34-10 It includes both the interrupt mask bits and channel configuration bits.

The interrupt mask provides bits for enabling/disabling each event defined in the interrupt circular table entry. Other bits provide various channel configuration options.

Offset: 0x18 Access: User read-write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MODE0	—	OCT	SUERM	FISU	—	UN	TXB	—	AERM	NID	IDL	MRF	RXF	BSY	RXB
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MODE1	POL	1	IDLM	—						TS	RQN	NOF			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 34-10. Extended Channel Mode Register (ECHAMR) - SS7 Mode

ECHAMR fields are described in Table 34-12.

Table 34-12. ECHAMR Fields Description

Bits	Name	Description
0,16	MODE0 MODE1	00 Transparent mode 01 HDLC mode 10 Reserved 11 SS7 mode (This is the required bit setting for an MCC to perform SS7.)
1, 5, 8	0	Reserved, should be cleared during initialization.
2–4 6–7 9–15	INTMSK	Interrupt mask bits. These bits are used for enabling/disabling the reporting of each possible event defined in the interrupt circular table entry. See Section 3.6.1.1, “Interrupt Circular Table Entry.” 0 Disabled 1 Enable
17	POL	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The QUICC Engine block does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL is used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The CP clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
18	1	Reserved, must be set.
19	IDLM	Idle mode. 0 No idle patterns are transmitted between frames. After transmitting NOF+1 flags, the transmitter starts sending the data of the frame. If the transmission is between frames and the frame buffers are not ready, the transmitter sends flags until it can start transmitting the data. 1 At least one idle pattern is sent between adjacent frames. The NOF value shall be no smaller than the PAD setting, see TxBD. If NOF = 0, this is identical to flag sharing in SS7. Mode flags precede the actual data. When IDLM = 1, at least one idle pattern is sent between adjacent frames. If the transmission is 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 IDLE mode is selected and NOF = 1, the following sequence is sent: .....init value, FF flag data,..... The init value before the idle will be ones.
20–25	—	Reserved, should be cleared during initialization.
10	TS	Receive time stamp. If this bit is set, a 4 byte time stamp is written at the beginning of every data buffer that the BD points to. Also, the data buffer must start from an address equal to 8*n-4 (n is any integer larger than 0).
11–12	RQN	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0 01 Queue number 1 10 Queue number 2 11 Queue number 3

Table 34-12. ECHAMR Fields Description (continued)

Bits	Name	Description
13–15	NOF	Number of flags. NOF defines the minimum number of flags before frames: 000 - at least 1 flag 001 - at least 2 flags .... 111 - at least 8 flags

34.2.2.4.2 Signal Unit Error Monitor (SUERM)—SS7 Mode

The microcode maintains the signal unit error rate monitor as described in ITU-T Q.703 paragraph 10, and ANSI T1.111-1996 paragraph 10.

The microcode uses SUERM, N, N_cnt, D, D_cnt and T parameters for the leaky-bucket implementation of the SU error monitor.

- After every N octets received while in octet counting mode, SUERM is incremented and an interrupt request can be generated (SUERM) depending on the interrupt mask.
- After D error-free frames have been received, SUERM is decremented. SUERM will not be decremented below zero.
- If SUERM reaches T, the SUERM is cleared and an interrupt is generated.

34.2.2.4.3 SUERM in Japanese SS7

The Japanese SS7 uses a time interval to monitor errors. If an error is present, it checks every 24 ms.

- An error flag is set that indicates whether current frame is errored or not.
- For every JTRDelay an error flag is checked.
- If there is no error, decrement the counter SUERM by 1 (not below zero).
- If there is an error, increment the counter SUERM by D.
- If SUERM reaches T, the counter SUERM is cleared and a “signal unit error rate monitor” interrupt is generated.

Table 34-13. Parameter Values for SUERM in Japanese SS7

Parameter	Definition	Value
T	Threshold	285
D	Upcount	16
JTRDelay	Length of interval (24ms)	0x2F

34.2.2.4.4 SS7 Configuration Register—SS7 Mode

The SS7 configuration register, shown on Figure 34-11 contains additional SS7 parameters.

Offset: 0x4E Access: User read-only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	—				AERM	SUERM_DIS	STD	SF_DIS	SU_FIL	SEN_FIS	O_ORN	O_ITUT	FISU_PAD			
W																
Reset																

Figure 34-11. SS7 Configuration Register (SS7_OPT)

Table 34-14 describes SS7 configuration register fields.

Table 34-14. SS7 Configuration Register Fields Description

Bits	Name	Description
0-3	—	Reserved, should be cleared during initialization.
4	**AERM**	Alignment error rate monitor enable. See Section 34.2.2.4.5, “AERM Implementation” 0 Do not enable AERM. 1 Enable AERM.
5	**SUERM_DIS**	Disable the SU error rate monitor. See Section 34.2.2.4.7, “Disabling SUERM” 0 Enable SUERM. 1 Disables both SUERM and AERM.
6	**STD**	Standard compliance 0 ITU-T/ANSI compliant 1 Japanese SS7 compliant
7	**SF_DIS**	Discard short frames (less than 5 octets) 0 Do not discard short frames. 1 Discard short frames.
8	SU_FIL	SU Filtering 0 Disable SU filtering. 1 Enable SU filtering.
9	**SEN_FIS**	Send FISU if first BD of frame is not ready. 0 Flags are sent if the current BD, which is the first BD of the frame, does not have its ready bit set. 1 FISUs are automatically sent if the current BD, which is the first BD of the frame, does not have its ready bit set.
10	**O_ORN**	Enter octet counting mode (OCM) on overrun. Should be cleared if using the Japanese standard. 0 Disable entering OCM if there are no receive BDs available. 1 Enter OCM if there are no receive BDs available. Note that when STD = 1, O_ORN = 1, and no receive buffers are ready, any received signal unit is treated as an errored signal unit.
11	**O_ITUT**	Enter octet counting mode (OCM) on ITU-T conditions (after an abort sequence or when an SU is too long). Should be cleared if using the Japanese standard. 0 Disable entering OCM on ITU-T conditions. 1 Enable entering OCM on ITU-T conditions.
12-15	**FISU_PAD**	Padding of the automatically transmitted FISUs. If the SEN_FISU bit is set, the CP will use the value of FISU_PAD as a number of pad character. Please refer to PAD parameter in Section 34.2.7.2, “Transmit Buffer Descriptor (TxBD)”

34.2.2.4.5 AERM Implementation

The SS7 microcode implements the ITU Q.703 alignment error rate monitor (AERM). The microcode uses the T, SUERM, M and M_cnt parameters. The M_cnt parameter is incremented for every T errored frames. If M_cnt reaches M, an AERM interrupt is generated to layer 3.

Note that in AERM mode no SUERM interrupt is generated. Also, the algorithm associated with D and D_cnt is disabled as per the ITU specification.

34.2.2.4.6 AERM in Japanese SS7

To meet the Japanese AERM requirements the user must change the parameters T and D. Note that the interrupt generated is not AERM but SUERM.

During proving, do the following:

1. Set SS7_OPT register to 0b0000 001X XX00 XXXX. The value of X doesn’t matter because these bits do not affect the operation of the error counter.
2. Clear JTRdelay parameter to'0.'
3. Set parameters T (threshold) and D (up counter) to'1.'
4. Clear parameter SUERM (error counter) to'0.'
5. Set JTTDelay to value required to generate 24ms delay.

These settings allow FISU or LSSU transmission to be delayed by the required 24ms (JTTDelay). They also allow the correct operation of the JT Q703 error counter and ensure that an SUERM interrupt is generated on the first SU received in error.

After proving period, set the parameters (T and D) to values according to the Japanese SUERM. See section Table 34-13.

To disable AERM and enter SUERM, do the following:

1. Set SUERM_DIS bit in SS7_OPT.
2. Set parameters (T, D & SUERM) for Japanese SUERM.
3. Clear SUERM_DIS bit in SS7_OPT.

34.2.2.4.7 Disabling SUERM

When SS7_OPT[SUERM_DIS] is set, the N_cnt and D_cnt parameters are not decremented and no SUERM interrupt is generated. This allows these parameters to be updated, for example, at the end of the proving period in alignment error monitoring.

Note: If the SS7 controller is in the octet counting mode (OCM) when SUERM_DIS is set, and consequently if no idles (only flags/data) are received, then after the host updates N_cnt and D_cnt and clears SUERM_DIS, the receiver will start in OCM. However if SUERM_DIS is set while in OCM and consequently idles are received, then the OCM state is left as-is.

34.2.2.4.8 SU Filtering—SS7 Mode

To reduce the overhead to the user software, a filtering algorithm has been adopted to allow superfluous frames to be discarded. This algorithm compares the first 3–5 bytes (depending on the type) of the current FISU or LSSU to the last SU received and discards the current SU if it has already been received.

34.2.2.4.9 Comparison Mask

A user programmable 5-byte mask exists in the parameter RAM map. When an SU is received, the controller checks the contents of the LI field. If LI is between 0 and 2, the SU (except for the CRC portion) will be masked according to the user programmable mask and will then be compared to the last SU received. The state machine for the matching algorithm is in Section 34.2.2.4.10, "Comparison State Machine."

The Mask1 and Mask2 channel-specific parameters construct the 5-byte user mask. The exact format and byte ordering are shown on Figure 34-12 and Figure 34-13.

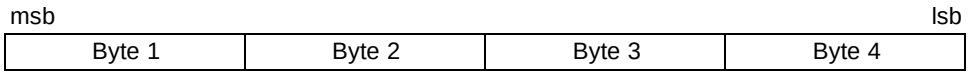


Figure 34-12. Mask1 Format

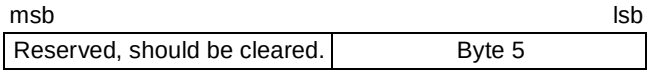
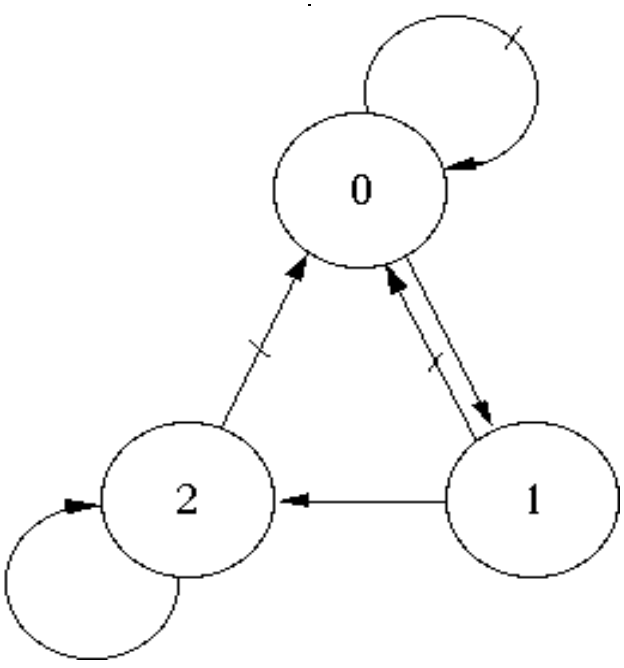


Figure 34-13. Mask2 Format

34.2.2.4.10 Comparison State Machine

The following state machine exists for filtering.



- State 0—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, go to State 1, else remain in State 0. The current SU will be received into a buffer descriptor.
- State 1—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, go to State 2, else go to State 0. The current SU will be received into a buffer descriptor.
- State 2—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, the current SU will be discarded (unless there is an error), the channel will remain in state 2 and SU error monitor will be adjusted accordingly. If the frames do not match, the current SU will be received into a buffer descriptor and the channel will return to State 0.

34.2.2.4.11 Filtering Limitations

Because the algorithm is purely checking identical SUs, two FISUs will be received after each MSU rather than merely one, even though they have the same sequence numbers.

Reception of an MSU resets the filtering algorithm. Also, reception of a short frame resets the filtering algorithm when SS7_OPT[SF_DIS] = 0; however, when SS7_OPT[SF_DIS] = 1 (short frames are discarded), the filtering algorithm remains unchanged.

34.2.2.4.12 Resetting the SU Filtering Mechanism

This command resets the filtering algorithm to ensure that the next SU will be received, even if it would normally have been filtered. This command could be issued periodically so that the CPU can check to make sure that the link is really up and not simply receiving flags.

To issue this MCC command, refer to Section 20.3.1, “QUICC Engine Command Register (CECR).” Use opcode 0b01_0001.

NOTE

For the MPC82*xx*, the opcode command for the RESET SU FILTER is 0b1110 (0xE). The QUICC Engine opcode command for the RESET SU FILTER is opcode 0b01_0001.

34.2.2.4.13 Octet Counting Mode—SS7 Mode

When entering the octet counting mode (OCM), the QUICC Engine block will load the user defined N register to its internal octet counter. While in the octet counting mode the RISC will decrement its internal counter for every unstuffed octet received. When the internal counter is decremented to zero, the QUICC Engine block will increment the SUERM register and reload the N register into the internal count register. In addition an interrupt (OCT) might be generated depending on the interrupt mask. The SS7 controller will enter octet counting mode under the following circumstances:

- An ABORT character is received at any time and SS7_OPT[O_ITUT] is set.
- The SU currently being received has exceeded the length programmed in the MFLR register and SS7_OPT[O_ITUT] is set.

- The receiver overruns and SS7_OPT[O_ORN] is set. Note that when no receive buffers are available, only octets are counted; that is, D_cnt is not decremented after receiving the frame.

The SS7 controller will leave octet counting mode when a valid signal unit is detected (with a valid CRC and a length less than MFLR and greater than 4).

NOTE

Octet counting mode applies only to the ITU-T and ANSI standards. The SS7 microcode will not work if both the Japanese standard and OCM features are selected.

34.2.3 Channel Extra Parameters

In addition to the information kept in the channel-specific parameter ram, a channel also has a set of pointers used to index its transmit and receive buffer descriptors. This information is kept in a set of channel-extra parameters. Table 34-15 describes the channel-extra parameters. These parameters are indexed using the channel number, as described in the table.

Table 34-15. Channel Extra Parameters

Offset[1]	Name	Width	Description
0x00	**TBASE**	Hword	TxBD base address. Offset of the channel's TxBD table relative to the MCCBASE (The base address of the BD table for this channel MCCBASE+8*TBASE)
0x02	**TBPTR**	Hword	TxBD pointer. Offset of the current BD relative to the MCCBASE. TBPTR is user-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel. (The address of the BD in use for this channel MCCBASE+8*TBPTR)
0x04	**RBASE**	Hword	RxBD base address. Offset of the channel's RxBD table relative to the MCCBASE. (The base address of the BD table for this channel MCCBASE+8*RBASE)
0x06	**RBPTR**	Hword	RxBD pointer. Offset of the current BD relative to the MCCBASE. RBPTR is user-initialized to RBASE before enabling the channel or after a fatal error before reinitializing the channel. (The address of the BD in use for this channel MCCBASE+8*RTBPTR)

[1] The offset relative to multi-port RAM base address + XTRABASE + 8*CH_NUM

34.2.4 Superchannel Table

The SCT serves as a mapping between the hardware resources being used as part of a superchannel (as programmed in SIRAM) and which channel's parameters are being used to manage that superchannel. When the SI encounters an SIRAM entry that is programmed to be part of a superchannel, the channel number in that SIRAM entry indicates to the MCC which hardware to use during that timeslot. However, when that hardware requires service (more data or request handling) from the QUICC Engine block, a lookup occurs using the Superchannel Table (SCT). As a result, there are multiple channels inside the MCC hardware to enhance performance, but a single channel as far as the user is concerned. This single high performance channel is what is defined as a "superchannel."

The MCC channel number used in the MCSEL field of the superchanneled SIRAM entry is used as an offset to a superchannel table entry that contains the MCC channel number whose parameters and buffer descriptors are being used to control that superchannel (see Figure 34-14). The only entries in the

superchannel table which must be initialized are those whose numbers appear in superchanneled entries in the SIRAM.

Note that no SCT is used for the receive side and the channel numbers used when programming receive SIRAM timeslots should always be that of the actual intended MCC receive channel FIFO.

	0	1	2 … 9	10	11	12	13	14	15
Field	0	0	**SuperChannel Number**	0	0	0	0	0	0
Addr	**MURAM_base_address+SCTPBASE**+2*(associated MCC channel_number) (MCC_channel _number is the number written in the MCSEL field of the corresponding SIRAM entry)								

Figure 34-14. Superchannel Table Entry

34.2.4.1 Transparent Slot Synchronization

Transparent slot synchronization (TSS) is used to ensure that data transmission and reception for a transparent superchannel begins on the intended timeslot of that superchannel. It is not required that the first timeslot that appears in the SIRAM programming for a transparent superchannel be the first to send or receive when the superchannel begins operation. The user indicates which timeslot in a superchannel should be the first to send or receive by programming the CNT and BYT fields of the superchanneled timeslots as described in “Programming SIx RAM Entries.”

34.2.4.2 Superchannelling Programming Examples

The example in Figure 34-15 shows the SI RAM programming and the super-channel table for two different transmitter superchannels running on the same TDM interface. One superchannel includes TDM timeslots 1, 6, and 7, which also happen to be programmed to use MCC channel numbers 1, 6, and 7. The second superchannel in this example is comprised of timeslots 2, 3, and 4 using MCC channels 2, 3, and 4. This approach of using a channel number which is the same as the timeslot number is arbitrary and not a requirement.

Note that entries in the superchannel table for MCC channels 1, 6 and 7 all are programmed to point to the channel-specific and channel-extra parameters for channel 1. Superchannel table entries for MCC channels 2, 3 and 4 are programmed such that these channels are managed by the parameters of channel 2.

SI RAM

0	1	2	3–10	11–13	14	15
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST
SI RAM Address						
1	0	0	0x0	0x1	1	0
1	0	1	0x1	0x0[1]	1	0
1	0	1	0x2	0x0[1]	1	0
1	0	1	0x3	0x7[2]	0	0
1	0	1	0x4	0x7[2]	0	0
1	0	0	0x5	0x1	1	0
1	0	1	0x6	0x7[2]	0	0
1	0	1	0x7	0x7[2]	0	0
1	0	0	0x8	0x1	1	1

1 First slot of the superchannel

2 Regular (not first) slot of the superchannel

Superchannel Table

	0–1	2–9	10–15
		CHANNEL NO	
MURAM_Base + SCTPBASE +			
0x0		—	
0x2		0x1	
0x4		0x2	
0x6		0x2	
0x8		0x2	
0xA		—	
0xC		0x1	
0xE		0x1	
0x10		—	

The superchannel BD tables are associated with channels 1 and 2 (no BD tables are necessary for channels 3, 4, 6, and 7)

Figure 34-15. Transmitter Superchannel Example

In this example, data is expected to be sent on the first timeslots allocated for each superchannel. Thus for the first superchannel, timeslot 1 has CNT=0 and BYT=1, the “first byte” condition and the remaining timeslots that are part of this superchannel—timeslots 6 and 7, have CNT=0x7 and BYT=0. This indicates to the MCC that when this transparent superchannel becomes active it should begin sending data on timeslot 1. If the application required that data not be sent on this superchannel until timeslot 7, for example, then timeslots 1 and 6 have CNT=0x7 and BYT=0 and timeslot 7 would be programmed with CNT=0 and BYT=1.

Similarly, the second superchannel in this example sends data beginning with its first timeslot, timeslot 2. It contains the “first byte” condition of CNT=0 and BYT=1. If the application required a different timeslot in this superchannel, either timeslot 3 or 4, that channel could be programmed to have the “first byte” condition instead.

Figure 34-16 shows the SIRAM programming for transparent receiver superchannels which use slot synchronization. This example assumes a timeslot configuration similar to the transmit example in Figure 34-15. For this receive example, to guarantee that reception begins on the first timeslots for each superchannel, all timeslots that correspond to superchannels are programmed as superchanneled timeslots and the first timeslot for each superchannel is programmed with the “first byte” CNT and BYT conditions.

Note that the receive examples do not include a superchannel table because a superchannel table is only used on the transmit side. Receive SIRAM entries should always be programmed using the channel number of the managing MCC channel for that superchannel (the same MCC channel number used in the superchannel table entries corresponding to the transmit channels for that superchannel). The MCC does not share the individual data channel's hardware resources on the receive side.

SI RAM

0	1	2	3–10	11–13	14	15	
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST	
SI RAM Address							
1	0	0	0x0	0x1	1	0	Regular Channel
1	0	1	0x1	0x0[1]	1	0	Superchannel 1
1	0	1	0x2	0x0[1]	1	0	Superchannel 2
1	0	1	0x2	0x7[2]	0	0	Superchannel 2
1	0	1	0x2	0x7[2]	0	0	Superchannel 2
1	0	0	0x3	0x1	1	0	Regular Channel
1	0	1	0x1	0x7[2]	0	0	Superchannel 1
1	0	1	0x1	0x7[2]	0	0	Superchannel 1
1	0	0	0x4	0x1	1	1	Regular Channel

1 First slot of the superchannel

2 Regular (not first) slot of the superchannel

The superchannel BD tables are associated with channels 1 and 2

Figure 34-16. Receiver Superchannel with Slot Synchronization Example

Figure 34-17 shows the SIRAM programming for the same overall configuration as the previous examples, but in this case it does not matter to the application on which timeslot of a superchannel that reception begins. Thus, slot synchronization is not necessary, the timeslots do not need to be programmed as superchanneled timeslots, and the CNT and BYT fields can be programmed normally.

SI RAM **SI RAM**

0	1	2	3–10	11–13	14	15	
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST	
SI RAM Address							
1	0	0	0x0	0x1	1	0	Regular Channel
1	0	0	0x1	0x1	1	0	Superchannel 1
1	0	0	0x2	0x1	1	0	Superchannel 2
1	0	0	0x2	0x1	1	0	Superchannel 2
1	0	0	0x2	0x1	1	0	Superchannel 2
1	0	0	0x3	0x1	1	0	Regular Channel
1	0	0	0x1	0x1	1	0	Superchannel 1
1	0	0	0x1	0x1	1	0	Superchannel 1
1	0	0	0x4	0x1	1	1	Regular Channel

Note: The superchannel BD tables are associated with channels 1 and 2.

Figure 34-17. Receiver Superchannel without Slot Synchronization Example

34.2.5 MCC Configuration Registers (MCCF)

The MCC configuration register (MCCF), shown in Figure 3-18, defines the mapping of the MCC channels, paired with the serial interface (SI), to the TDM channels. For the MCC-SI, each of the four 32-channel subgroups can be connected to one of the eight TDM highways (TDMA, TDMB, TDMC, TDMD,TDME, TDMF, TDMG, TDMH).

Offset: 0x08 Access: User read-write

	0–2	3	4–6	7	8–10	11	12–14	15	16–18	19	20–22	23	24–26	27	28–30	31
R W	Group 1	0	Group 2	0	Group 3	0	Group 4	0	Group 5	0	Group 6	0	Group 7	0	Group 8	0
Reset	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0	0 0 0	0

Figure 34-18. SI MCC Configuration Register (MCCF)

NOTE_on backward compatibility

The configuration register is not backward compatible. The QUICC Engine MCC supports 8 TDMs whereas the CPM supports 4 TDMs. As a result, the number of bits needed for the QUICC Engine MCCs configuration was increased from 2 to 4.

Table 34-16 describes MCCF fields.

Table 34-16. MCCF Field Descriptions

Bits	Name	Description
0–2, 4–6, 8–10 12–14, 16–18, 20-22, 24–26,28-30	**GROUP *x***	Group x of channels is used by TDM y as shown in Table 3-19. 000 Group x is used by TDM A. 001 Group x is used by TDM B. 010 Group x is used by TDM C. 011 Group x is used by TDM D. 100 Group x is used by TDM E. 101 Group x is used by TDM F. 110 Group x is used by TDM G. 111 Group x is used by TDM H.
3,7,11,15,19,23,27,31	**Reserved**	Must be set to 0.

Table 34-17 describes group assignments.

Table 34-17. Group Channel Assignments

Group	Channels
Group1 in MCCF 1	0–31
Group2 in MCCF1	32–63
Group3 in MCCF1	64–95
Group4 in MCCF1	96–127
Group5 in MCCF 1	128–159
Group6 in MCCF1	160–191
Group7 in MCCF1	192–223
Group8 in MCCF1	224–255

NOTE

The TDM group channel assignments made in MCCF must be coherent with the SI register programming and SI RAM programming.The user must also program MCCF before enabling the TDM channel in the SIGMR.

34.2.6 MCC Exceptions

The MCC interrupt reporting scheme has two levels. The circular interrupt tables (illustrated in Figure 34-19) report channel-specific events and are masked by each channel's INTMSK field located in channel-specific parameter RAM. The MCCE event register (described in Section 34.2.6.1, "MCC Event Register (MCCE)/Mask Register (MCCM)") reports some global-level events and whether new activity has taken place in any of the MCC's interrupt tables. These events can be masked by the MCCM.

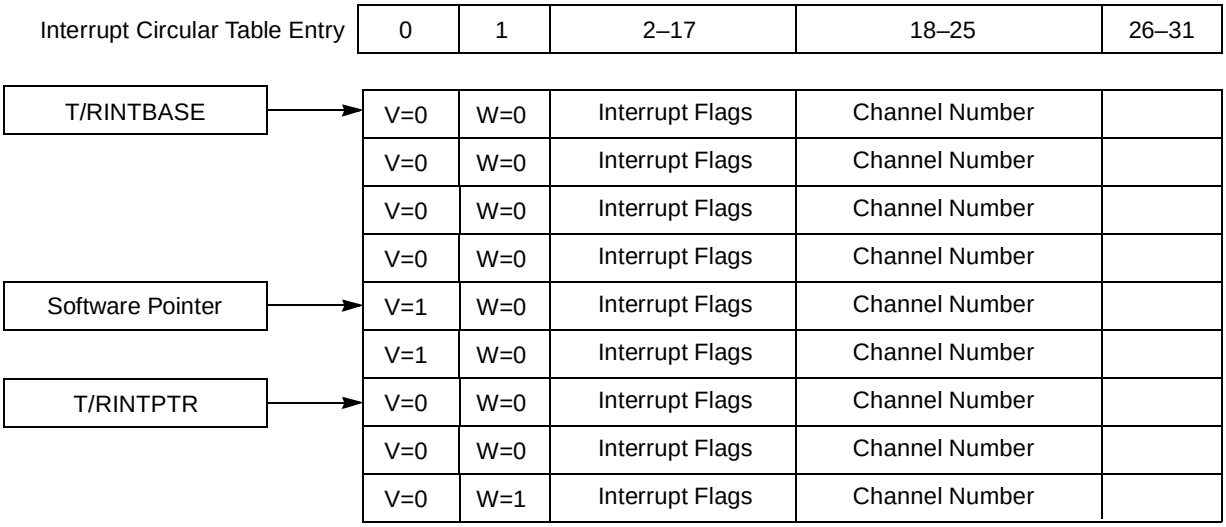


Figure 34-19. Interrupt Circular Table

There is one table for transmitter interrupts and from one to four tables for receiver interrupts. Each channel is programmed to report receiver interrupts in one of the receiver tables. This way receiver interrupts can be sorted, for example, by priority. Each interrupt circular table must be least two entries long.

T/RINTBASE and T/RINTPTR, which are user-initialized global MCC parameters (See Table 34-2"), point to the starting location of the table (in external memory) and the current empty position (initialized at the top of the table) available to the QUICC Engine block. All the entries in the table must be user-initialized with 0x00000000, except for the last one which must be initialized with 0x40000000 (W = 1, thus defining the end of the table). When an MCC channel generates an interrupt request, the QUICC Engine block writes a new entry to the table (with V = 1) and increments T/RINTPTR (if W = 1 for the current entry, T/RINTPTR is loaded with T/RINTBASE).

The circular interrupt tables consist of channel-specific events, with a bit for each possible event as well as the number of the channel reporting that event. Each channel has an INTMSK field that determines which events on that particular channel trigger the creation of a new entry in the interrupt tables. Whenever a new entry is added to an interrupt table, the MCC will set the appropriate TINT or RINTx bit in the MCCE event register, if that bit is properly enabled in MCCM mask register. If there was no room in the interrupt table for a new entry, the corresponding queue overflow bit (QOVx) will be set in the MCCE and the interrupt information is lost, although operation will continue.

After an MCC interrupt reaches the core, the software should read the corresponding MCCE. After clearing the appropriate event bits by writing ones to them, the software may begin processing the table(s) that contain pending events, as indicated by the bits MCCE[RINTx] and MCCE[TINT]. When processing the interrupt tables, the software must clear each entry's valid bit (V) (see Section 34.2.6.2, "Interrupt Circular Table Entry"). The user follows this procedure until it reaches an entry with V = 0. It may not be appropriate for an application to process every new entry of all interrupt tables at once, depending on desired interrupt handler latency or other factors. It is up to the user to determine an interrupt handling scheme that provides desired performance and functionality.

34.2.6.1 MCC Event Register (MCCE)/Mask Register (MCCM)

The MCC event register (MCCE) is used to report events and generate interrupt requests. For each of its flags, a programmable mask/enable bit in MCCM determines whether an interrupt request is generated. The MCC mask register (MCCM) is used to enable/disable interrupt requests. For each flag in the MCCE there is a programmable mask/enable bit in MCCM which determines whether an interrupt request is generated. Setting an MCCM bit enables and clearing an MCCM bit disables the corresponding interrupt.

MCCE bits are cleared by writing ones to them; writing zeros has no effect.

Figure 3-20 shows MCCE and MCCM bits.

Offset: Access: User read-write

	0	1	2	3	4	5	6	7	8			11	12	13	14	15
R W	QOV0	RINT0	QOV1	RINT1	QOV2	RINT2	QOV3	RINT3	—				TQOV	TINT	GUN	GOV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 34-20. MCC Event Register (MCCE)/Mask Register (MCCM)

Table 34-18 describes MCCE fields.

Table 34-18. MCCE/MCCM Register Field Descriptions

Bits	Name	Description
0	QOV0	QOVx—Receive interrupt queue overflow. IQOV is set (and an interrupt request generated) by the QUICC Engine block whenever an overflow occurs in the transmit circular interrupt table. This occurs if the QUICC Engine block tries to update an interrupt entry that was not handled by the user (such an entry is identified by V = 1). RINTx—Receive interrupt. When RINT = 1, the MCC generated at least one new entry in the receive interrupt circular table. After clearing it, the user reads the next entry from the receive interrupt circular table and starts processing a specific channel's exception. The user returns from the interrupt handler when it reaches a table entry with V = 0.
1	RINT0	
2	QOV1	
3	RINT1	
4	QOV2	
5	RINT2	
6	QOV3	
7	RINT3	
8–11	—	Reserved, should be cleared.
12	TQOV	Transmit interrupt queue overflow. TQOV is set (and an interrupt request is generated) by the QUICC Engine block whenever an overflow occurs in the transmit circular interrupt table. This condition occurs if the QUICC Engine block attempts to write a new interrupt entry into an entry that was not handled by the user. Such an entry is identified by V = 1.

Table 34-18. MCCE/MCCM Register Field Descriptions (continued)

Bits	Name	Description
13	TINT	Transmit interrupt. When TINT = 1, at least one new entry in the transmit interrupt circular table was generated by MCC. After clearing it, the user reads the next entry from the transmit interrupt circular table and starts processing a specific channel's exception. The user returns from the interrupt handler when it reaches a table entry with V = 0.
14	GUN	Global transmit underrun. This flag indicates whether an underrun occurred inside the MCC's transmit channel array. The user must clear this bit.
15	GOV	Global receiver overrun. This flag indicates whether an overrun occurred inside the MCC's receive channel array. The user must clear this bit.
16–31	—	Reserved, should be cleared.

34.2.6.2 Interrupt Circular Table Entry

Each interrupt circular table entry, shown in Figure 34-19, contains information about channel-specific events. The transmit circular table shows only events caused by transmission; the receive circular tables shows only events caused by reception. The corresponding interrupt mask bits are mode-dependent; refer to the appropriate section:

- Section 34.2.2.1.2, "Interrupt Mask (INTMSK)—HDLC Mode
- Section 34.2.2.2.2, "Interrupt Mask (INTMSK)—Transparent Mode
- Section 34.2.2.3.3, "Interrupt Circular Table Entry and Interrupt Mask (INTMSK)—AAL1 CES
- Section 34.2.2.4.1, "Extended Channel Mode Register (ECHAMR)—SS7 Mode

Offset 0xC_0040 Access: User read-only

:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	**V**	**W**	OCT[1] / —	SUERM[1] / —	FISU[1] / SLIPE[1]	— / SLIPS[2]	UNTXB		—	AERM[1] / —	NID	IDL	MRF	RXF	BSY	RXB
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

1 SS7 mode only. Otherwise, reserved.

2 Only used in conjunction with AAL1 CES

Figure 34-21. Interrupt Circular Table Entry

Table 34-19 describes interrupt circular table fields.

Table 34-19. Interrupt Circular Table Entry Field Descriptions

Bits	Name	Description
0	**V**	Valid bit. V = 1 indicates that this entry contains valid interrupt information. Upon generating a new entry, the QUICC Engine block sets V = 1. The user clears V immediately after it reads the interrupt flags of the entry, prior to processing the interrupt. The V bits in the table are user-initialized. During initialization, the user must clear those bits in all table entries.
1	**W**	Wrap bit. W = 1 indicates the last interrupt circular table entry. The next event's entry is written/read (by RISC/user) from the address contained in INTBASE (see Table 34-2). During initialization, the user must clear all W bits in the table except for the last one which must be set.
2	—	Reserved, should be cleared. (If SS7 mode, refer to the following descriptions.)
	OCT	SS7 mode only: N octets received. If the channel is in octet counting, this bit is set when N octets have been received.
3	—	Reserved, should be cleared. (If SS7 mode, refer to the following descriptions.)
	SUERM	SS7 mode only: SU error monitor threshold reached. The SU error monitor has reached the programmed threshold T.
4	FISU	SS7 mode only: FISU transmission started. The CP has started automatic FISU transmission if the first BD of frame does not have its ready bit set and the SEN_FISU bit is enabled in SS7_OPT register. Please refer to SEN_FISU bit in Section 3.2.4.3, "SS7 Configuration Register—SS7 Mode.
	SLIPE	Only used in conjunction with AAL1 CES. Slip End. Set when an MCC channel interworking with an ATM channel exits the slip state (the connection's CESAC falls to the MCC_Start threshold). At this point, the transmitter stops sending the underrun template (or last buffer) and starts sending valid data.
5	—	Reserved, should be cleared.
	SLIPS	Only used in conjunction with AAL1 CES.Slip Start. Set when an MCC channel interworking with an ATM channel enters a slip state (the channel's CESAC reaches the MCC_Stop threshold). At this point the transmitter freezes and begins sending the underrun template (or last buffer) until CESAC falls to the MCC_Start threshold.
6-7	UNTXB	Underrun or Tx Buffer 00 No event 01TXB 10 UN 11 Channel underrun
8	—	Reserved, should be cleared.
9	—	Reserved, should be cleared. (If SS7 mode, refer to the following description.)
	AERM	SS7 mode only: Alignment error rate monitor threshold (M value in SS7 channel-specific parameters) has been reached.
10	NID	Set whenever a pattern that is not an idle pattern is identified.
11	IDL	Idle. Set when the channel's receiver identifies the first occurrence of idle (0xFFFE) after any non-idle pattern.
12	MRF	Maximum receive frame length violation. This interrupt occurs when more bytes are received than the value specified in MFLR. This interrupt is generated as soon as the MFLR value is exceeded; the remainder of the frame is discarded
13	RXF	Rx frame. A complete HDLC frame has been received.
14	BSY	Busy. A frame was received but was discarded due to lack of buffers.

Table 34-19. Interrupt Circular Table Entry Field Descriptions (continued)

Bits	Name	Description
15	RXB	Rx buffer. A buffer has been received on this channel that was not the last buffer in frame. This interrupt is also given for different error types that can happen during reception. Error conditions are reported in the RxBD.
16–17	—	Reserved, should be cleared.
18–25	CN	Channel number. Identifies the requests channel index (0–255).
26–31	—	Reserved, should be cleared.

34.2.7 MCC Buffer Descriptors

Each MCC channel requires two BD tables (one for transmit and one for receive). Each BD contains key information about the buffer it defines. The BDs are accessed by the MCC as needed. BDs can be added dynamically to the BDs chain. The RxBDs chain must include at least two BDs; the TxBD chain must include at least one BDs.

The MCC BDs are located in external memory.

34.2.7.1 Receive Buffer Descriptor (RxBD)

Figure 34-22 shows the RxBD.

	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	SF[1]	—
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 34-22. MCC Receive Buffer Descriptor (RxBD)

[1] SS7 mode only. Otherwise, reserved.

RxBD fields are described in Table 34-20.

Table 34-20. 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 QUICC Engine block 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 QUICC Engine block. When E = 1, the user should not write any fields of this RxBD.
1	—	Reserved, should be cleared.

Table 34-20. RxBD Field Descriptions (continued)

Bits	Name	Description
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 block 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.
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 QUICC Engine block after this BD is closed). 1 The empty bit (bit 0) is not cleared by the QUICC Engine block after this BD is closed, allowing the associated data buffer to be overwritten automatically when the QUICC Engine block 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 double 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 non octet-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 invalid byte is always eliminated from the frame data. Data size reflects only the valid data bytes. A Non octet frame which is smaller than 1 byte is eliminated completely.
12	AB	Rx abort sequence. A minimum of seven consecutive ones 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.

Table 34-20. RxBD Field Descriptions (continued)

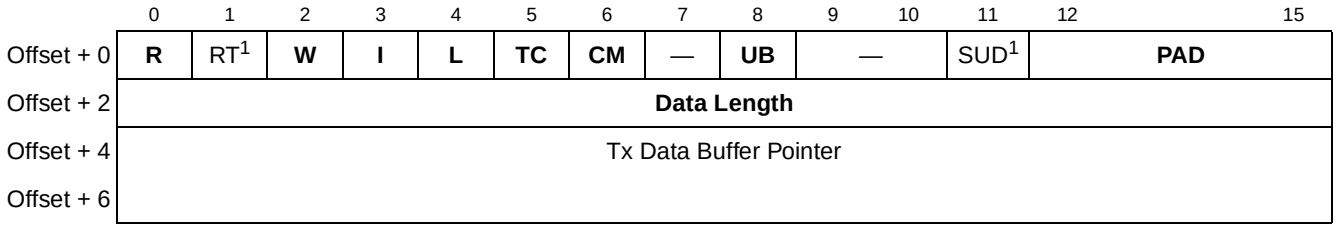
Bits	Name	Description
13	CR	Rx CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14	—	Reserved, should be cleared.
	SF	SS7 mode only: Short frame indication. Set if the received frame is less than 5 octets.
15	—	Reserved, should be cleared.

The data length and buffer pointer are described as follows:

- Data length. Data length is the number of octets written by the QUICC Engine block into this BD's data buffer. It is written by the QUICC Engine block when the BD is closed. When this is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including two or four bytes for CRC). Note that memory allocated for buffers should be not smaller than the contents of the maximum receive buffer length register (MRBLR). The data length does not include the time stamp.
- Rx buffer pointer. The receive buffer pointer points to the first location of the associated data buffer. This value must be equal to 8*n if CHAMR[TS] = 0 and equal to 8*n - 4 if CHAMR[TS] = 1 (where n is any integer larger than 0).

34.2.7.2 Transmit Buffer Descriptor (TxBD)

Figure 34-23 shows the TxBD.



[1] SS7 mode only. Otherwise, reserved.

Figure 34-23. Transmit Buffer Descriptor (TxBD)

Table 34-21 describes TxBD fields.

Table 34-21. TxBD Field Descriptions

Bits	Name	Description
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 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 is ready to be transmitted. The transmission may have begun, but it has not completed. The user cannot modify this BD once this bit is set.

Table 34-21. TxBD Field Descriptions (continued)

Bits	Name	Description
1	RT	SS7 mode only: Retransmit. 0 Normal operation 1 The QUICC Engine block repeats transmission of this BD until the RT bit is cleared. After the RT bit is cleared the QUICC Engine block advances to the next BD in the table. This feature is useful for automatic LSSU retransmission. **Note:** This bit is reserved in all other modes of operation.
2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the RISC receives incoming data into the first BD in the table (the BD pointed to by TBASE). The number of TxBDs in this table is programmable and determines which BD should contain the wrap bit.
3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the circular interrupt circular table entry is set when this buffer has been serviced by the MCC. This bit can cause an interrupt, if enabled.
4	**L**	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
5	**TC**	Tx CRC. 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 by sending 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 QUICC Engine block does not clear the ready bit after this BD is closed, allowing the associated data 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 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.
9–10	—	Reserved, should be cleared.
11	—	Reserved, should be cleared.
	SUD	SS7 mode only: Signal unit delay 0 This buffer does not have a transmission delay. 1 A time delay of JTTDelay x 512 µs passes before this buffer is transmitted. Can be used for LSSU transmission according to the JT Q.703 Standard which defines a 24 ms delay between back-to-back LSSUs. This bit is only valid when SS7_OPT[STD] is set.
12–15	**PAD**	Pad characters. These four bits indicate the number of PAD characters (0x7E or 0xFF depending on the IDLM mode selected in the CHAMR register) that the transmitter sends after the closing flag.

The data length and buffer pointer are described below:

- Data length. The data length is the number of bytes the MCC should transmit from this BD’s data buffer. It is never modified by the RISC. The value of this field should be greater than zero.

- Tx buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer must reside in external memory. This value is never modified by the RISC.

34.2.8 MCC Emergency Request Level (MERL)

The MCC offers flexible options to determine when the MCC will go into emergency mode, providing system optimization.

Figure 34-24 shows the MCC Emergency Request Level (MERL) register.

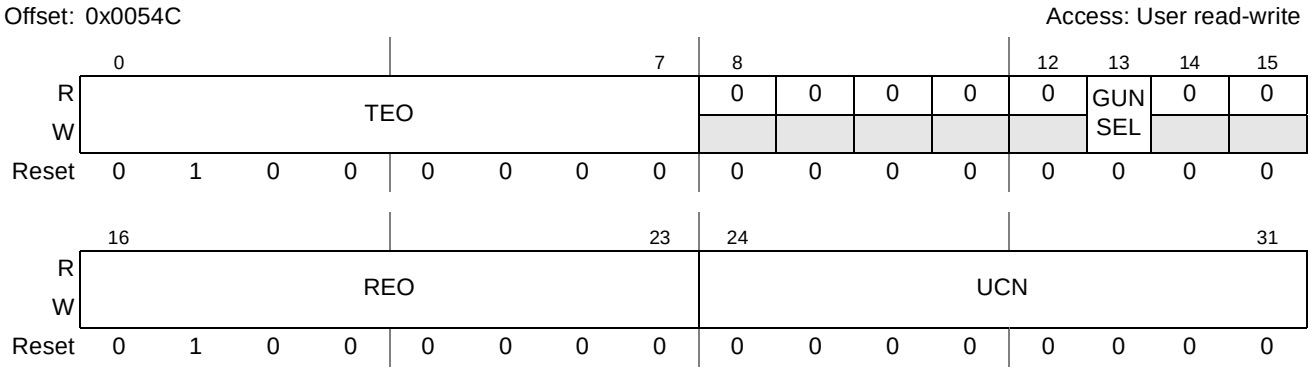


Figure 34-24. MCC Emergency Request Level (MERL)

Table 34-22 describes the MERL register.

Table 34-22. MCC Emergency Request Level (MERL)

Bits	Name	Description
0–7	TEO	Transmit Emergency Offset is an offset from the beginning of the transmitter request FIFO. The offset is actually the amount of data in the request FIFO. Emergency request will be asserted when the amount of data in the request FIFO equals the amount or more. The FIFO has 256 requests. There are 2 special cases. • The TEO is equal to 11111111- The MCC Tx will never work in emergency regime. • The TEO is equal to 00000000 - The MCC Tx will always be in emergency regime.
8–12	Reserved	Must be zero.
13	GUN SEL	GUN SEL- Global Underrun Select 0- GUN - If the transmitter has an underrun in one of it's channels the transmitter will transmit idles until reset. 1- An indication will be given as to the specific channel number that has an underrun. The line may see a period of idles, followed by old data, followed by idles again.
14–15	Reserved	Must be zero.

Bits	Name	Description
16–23	REO	Receive Emergency Offset is an offset from the beginning of the receive request FIFO. The offset is actually the amount of data in the request FIFO. Emergency request will be asserted when the amount of data in the request FIFO equals this amount or less. The FIFO has 256 requests. There are 2 special cases. • The REO is equal to 11111111- The MCC Rx will never work in emergency regime. • The REO is equal to 00000000 - The MCC Rx will always be in emergency regime.
24–31	UCN	Underrun Channel Number - During an underrun, the UCN will contain the most recent channel number that caused an underrun.

NOTE

The MERL provides the capability of raising the priority of the MCC in an application, and may therefore benefit from tweaking these settings with the system fully loaded to discover how to best optimize that specific system. The recommended initial setting values of this register is half the number of active channels in system. The default value is 64 which is the equivalent of a quarter of an E1 line of requests in each TDM.

34.3 MCC Commands

The user starts channels by writing to the TSTATE/RSTATE registers as described in Section 34.2.2.1.4, “Internal Receiver State (RSTATE)—HDLC Mode,” and Section 34.2.2.1.1, “Internal Transmitter State (TSTATE)—HDLC Mode.”

The following commands, used to stop and initialize channels, are issued to the MCC by writing to the CECR as described in “QUICC Engine Command Register (CECR).” All MCC channels must be initialized using one of the following commands before being used in an active TDM.

Table 34-23. MCC Commands

Command	Description
INIT RX AND TX[1]	Performs both INIT RX and INIT TX commands contiguously, using the channel number supplied with the command.
INIT RX[1]	Initializes MCC receive FIFOs in groups of 32 channels, the command initializes the group with the channel number programmed in the CECR[MCN] field when the command is issued. This command should only be issued when the channels are disabled. To initialize more than 32 channels, reissue the command with the appropriate channel numbers. The INIT TX AND RX command may be used to initialize both the receive and transmit sides of an MCC channel at the same time. Note that this command will initialize the first 16 channels to be preloaded to receive 8 bits, and the second set of channels will be preloaded to receive 16 bits. This is done to stagger when channels require servicing and spread out QUICC Engine block loading.
INIT TX[1]	Initializes MCC transmit FIFOs in groups of 32 channels, the command initializes the group with the channel number programmed in the CECR[MCN] field when the command is issued. This command should only be issued when the channels are disabled. To initialize more than 32 channels, reissue the command with the appropriate channel numbers. The INIT TX AND RX command may be used to initialize both the receive and transmit sides of an MCC channel at the same time. Note that this command will initialize the first 8channels to be preloaded with 16 bits of idle, and the second set of channels will be preloaded with 16 bits of idle. This is done to stagger when channels require servicing and spread out QUICC Engine block loading.

Table 34-23. MCC Commands

Command	Description
[2]INIT TX AND RX[1] (16 BITS)	Same as regular INIT RX AND TX, except that all channels are equally preloaded with idle. For applications in which all channels must begin sending or receiving data in the same TDM frame.
INIT TX, ONE[1] CHANNEL	Performs the INIT TX command but only for the channel number programmed in CECR[MCN] as opposed to initializing 32 channels at once.
INIT RX, ONE[1] CHANNEL	Performs the INIT RX command but only for the channel number programmed in CECR[MCN] as opposed to initializing 32 channels at once.
MCC RESET	Initializes the state machine hardware of the MCC indicated in CECR[PAGE] and CECR[SBC]. It has the same effect on the MCC block that a QUICC Engine block reset does. Required after the GUN or GOV MCC event occurs.
STOP TRANSMIT	Disables the transmission on the selected channel and clears CHAMR[POL]. When this command is issued in the middle of a frame, the QUICC Engine block sends an ABORT indication and then idles/flags on the selected channel. If this command is issued between frames, the QUICC Engine block sends only idles or flags (depending on CHAMR[IDLM]). TBPTR points to the buffer that the QUICC Engine block was using when the STOP TRANSMIT command was issued.
STOP RECEIVE	Forces the receiver of the selected channel to terminate reception. After this command is executed, the QUICC Engine block does not change the receive parameters in the multi-port RAM. The user must initialize the channel receive parameters in order to restart reception.

1 The INIT PARAMETERS style commands are also used to reset the MCC channel internal HW and these commands need to be issued to cover any channel number used, whether used normally or as part of a superchannel.

2 This init should be used when working with 16 bit entries for supper channel in order to avoid GUN or CUN.

34.4 MCC Initialization Information

The MCC must be initialized and started/stopped in relation with the corresponding TDMs. The following two sections present the initialization and start/stop sequences which must be followed for single channels and superchannels.

The following is a general sequence for initializing an MCC and its channels after reset:

1. Program the parallel I/O port interface for the TDM to be used, see Section 3.4, “Parallel I/O Ports.”.
2. Program the SIU’s interrupt controller to mask or enable MCC-related interrupts as desired, see Section 19.2, “Interrupt Controller.”
3. Program the SI’s SIRAM and related registers. If the user wishes to enable the TDM at this time, the SIRAM programming cannot yet contain MCC-related timeslots. Those timeslots should be NULL entries and not be programmed for MCC usage until after MCC-related initialization is complete. Refer to Chapter 36, “Serial Interface with Time-Slot Assigner,” for SI programming details.
4. Initialize buffer descriptors and data buffers as needed.
5. Initialize all MCC global parameters.
6. Initialize MCC channel-specific parameters for channels to be used. (Note that TSTATE and RSTATE can be fully programmed here to begin data transmission and reception as soon as the TDM is enabled. If a user wishes to wait until after the TDM is enabled before starting data

transmission and reception, the user may program only the FCR portions of RSTATE and TSTATE; and STOP TX and STOP RX commands should be issued for the appropriate channels before enabling the TDM.)

7. Initialize MCC channel-extra parameters for channels to be used.
8. Initialize Superchannel Table if superchannels are to be used.
9. Issue MCC INIT commands as appropriate to make sure all MCC FIFOs that are to be used are initialized.
10. Enable TDM (or if TDM is already enabled, the user may now reprogram the TDM to include MCC-related timeslots. This programming is done in the SI).
11. If the user did not program a channel's TSTATE and RSTATE in step 6 to begin data transmission and reception immediately upon having an active TDM, the user may program the start conditions at this time.

NOTE

Some steps may be performed out of order. However it is critical, regardless of their relative sequence, that steps 5, 6, 7, 8 and 9 occur before the enabling of the TDM or an active TDM is reprogrammed to include MCC-related timeslots. Failure to properly initialize MCC parameters and issue appropriate INIT commands before MCC-related timeslots are programmed in an active TDM may result in spurious GUN events or other anomalous behavior.

34.4.1 Stopping and Restarting a Single-Channel

The following sequence must be followed to stop a single channel in order to change the SI without using the shadow SI:

1. Issue a STOP command for the respective channel as described in Section 34.3, "MCC Commands," or change the associated SI RAM entry to point to a channel which is not active and wait for two frame periods in order to clear the internal FIFOs.
2. Change the channel parameters.
3. Enable the MCC channel(s) as described in Section 34.2.2.1.1, "Internal Transmitter State (TSTATE)—HDLC Mode," and Section 34.2.2.1.4, "Internal Receiver State (RSTATE)—HDLC Mode." or change the associated SI RAM entry to point to the respective channel.

It is possible to change the SI using the SI shadow while the channel is active. Both the primary and the shadow configuration of the SI RAM must observe the configuration defined in MCCF (see Section 34.2.5, "MCC Configuration Registers (MCCF)")". The MCCF cannot be changed while there are active channels.

The following sequence must be followed to stop a single channel in order to change the MCC parameters of the respective channel:

1. Issue a STOP command for the respective channel as described in Section 34.3, "MCC Commands,".
2. Change the SI.

3. Enable the MCC channel(s) as described in Section 34.2.2.1.1, “Internal Transmitter State (TSTATE)—HDLC Mode,” and Section 34.2.2.1.4, “Internal Receiver State (RSTATE)—HDLC Mode.”

34.4.2 Stopping and Restarting a Superchannel

The following sequence must be followed to stop a superchannel in order to change the SI:

1. Issue a STOP command for the respective channel as described in Section 34.3, “MCC Commands.”
2. Disable the TDM.
3. Change the SI.
4. Enable the TDM.
5. If necessary, change the MCC parameters (in MURAM and external memory).
6. Enable the MCC channel(s) as described in Section 34.2.2.1.1, “Internal Transmitter State (TSTATE)—HDLC Mode,” and Section 34.2.2.1.4, “Internal Receiver State (RSTATE)—HDLC Mode.”

Channels can’t be added dynamically to a Superchannel. The Superchannel must be deactivated, initialized with the new channels and only then reactivated. Shadow RAM maybe used to change the frame dynamically.

34.4.3 Global Underrun and Channel Underrun Recovery Routine

The MCC has 2 ways of handling a transmitter data underrun. The first is global underrun that views the lack of data as a critical error that must result in resetting the whole MCC. This is usually true if the underrun occurs due to performance issues.

The second way is channel underrun that handles only the specific channel that has no data. This is recommended only for systems with stable performance.

During a channel underrun, the Tx data line will be pulled to idles only during that channel’s period in the TDM line. The channel number will appear in MERL[16:23], see Section 34.2.8, “MCC Emergency Request Level (MERL).”

34.4.3.1 GUN RECOVERY

- Issue a RESET HOST command or a total RESET to the Device.
- Execute a full MCC initialization.

34.4.3.2 Channel Underrun RECOVERY

- Issue STOP TX to that channel.
- Execute the MCC initialization for that specific channel.
- Issue RESTART sequence.

34.4.4 Global Overrun Recovery Routine

Global overrun is a situation where there is overflow in the Rx data buffers resulting in data corruption. The recovery routine is similar to that of Global Underrun.

- Issue a RESET HOST command or a total RESET to the Device.
- Execute a full MCC initialization.

Chapter 35
ATM AAL1 Circuit Emulation Service

This chapter describes implementation of circuit emulation service (CES) using ATM adaptation layer type 1 (AAL1) on the QUICC Engine block and should be used as a supplement to Chapter 32, "ATM Controller AAL0, AAL1, and AAL5."

35.1 Features

The AAL1 CES features on the QUICC Engine block are as follows:

- AAL1
  - — Reassembly
    - – Reassembles PDU directly to external memory
    - – Supports partially filled cells (configurable on a per-VC basis)
    - – Sequence number (SN) protection (CRC-3 and parity) check
    - – Implements a 3-step SN algorithm
    - – Detects and handles lost or misinserted cell
    - – Maintains bit count integrity (dummy cell insertion)
    - – Dummy cell contents can be programmed by the user (per UCC)
    - – Pointer verification in structured AAL1 cell format
    - – Automatic synchronization using the structured pointer during reassembly
    - – SRTS (synchronous residual time stamp) gathering on every cycle (8 cells) in unstructured AAL1
    - – Statistics gathering on a per-VC basis:
      - – AAL1 valid cell count
      - – AAL1 lost cell count
      - – AAL1 misinserted cell count
      - – AAL1 pointer mismatch/parity error
      - – AAL1 Rx buffer pre-overflow
  - — Segmentation
    - – Segment PDU directly from external memory
    - – Partially filled cells support (configurable on a per-VC basis)
    - – Sequence number generation
    - – Sequence number protection (CRC-3 and even parity) generation
    - – Pointer generation during segmentation in structure AAL1 cell format

  - Clock recovery using external SRTS logic during reassembly in unstructured AAL1
  - Statistics gathering on a per-VC basis:
    - AAL1 Tx cell count
    - AAL1 Tx buffer underrun

- Circuit emulation service (CES)
  - ATM to TDM
    - Structured and unstructured data are transferred between the ATM and MCC automatically without CPU intervention
    - In case of pre-underrun, the MCC start sending the last frame or the user-defined underrun template. The MCC and ATM controller automatically perform slip control with no CPU intervention.
    - In case of pre-overrun, the ATM receiver discards incoming cells until the MCC transmitter empties enough buffers for the receiver to restart.
    - Supports common channel signaling (CCS)
    - Supports channel associated signaling (CAS)
      - Up to 4 (one per trunk) CAS blocks residing in internal RAM when in automatic CAS mode and up to 8 blocks when in core CAS modify mode
      - Supports Nx64 E1/T1 channels
      - CAS routing table on per-VC basis
      - Automatic un-packing of the CAS information during reassembly and updating of the internal CAS block
  - TDM to ATM
    - Structured and unstructured data are automatically transferred between the MCC and ATM controller without CPU intervention
    - Supports common channel signaling (CCS)
    - Supports channel associated signaling (CAS)
      - Up to 4 (one per trunk) CAS blocks residing in internal RAM when in automatic CAS mode and up to 8 blocks when in core CAS modify mode
      - Support Nx64 E1/T1 channels
      - CAS routing table on per-VC basis
      - Automatic packing of CAS information from internal RAM to AAL1 Tx cells
      - Once per superframe, the QUICC Engine block fetches the CAS information from an external framer and updates each internal CAS block.

35.2 AAL1 CES Transmitter Overview

The QUICC Engine block supports both structured and unstructured AAL1 cell formats. For unstructured format, the transmitter reads 47 bytes from the external buffer and inserts them into the AAL1 user data field. For structured format, the transmitter reads 46 or 47 bytes from the external buffer and inserts them into the AAL1 user data field.

35.2.1 Data Path

The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is generated and inserted into the AAL1 Tx cell, as shown in Figure 35-1.

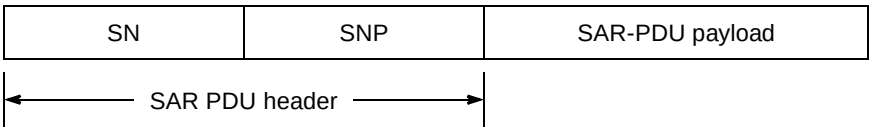


Figure 35-1. AAL1 Transmit Cell Format

When the transmitter operates in structured data transfer (SDT) mode, two types of AAL1 cells are defined: P (pointer) format and non-P format. The two formats are shown below.

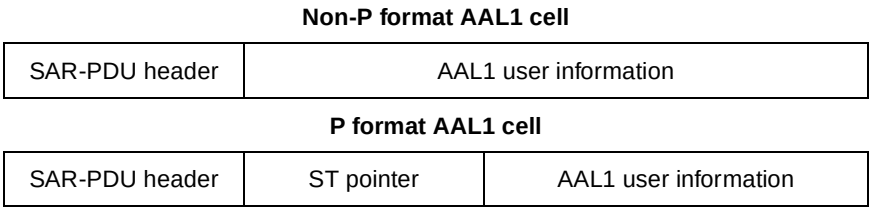


Figure 35-2. AAL1 SDT Cell Types

The transmitter generates the structured pointer according to the I.363.1 ITU standard and inserts the pointer exactly once every cycle (eight successive cells). The transmitter will insert the structured pointer, at the first opportunity, into a cell with an even sequence count (SC). When the end of the structure is not present in the current cycle, a P-format cell with a dummy pointer (127) is inserted into the last cell (SC=6) of the cycle.

The QUICC Engine block supports partially filled cells configured on a per-VC basis. In this mode, only the valid octets are filled with user information; the rest of the cell is filled with padding octets.

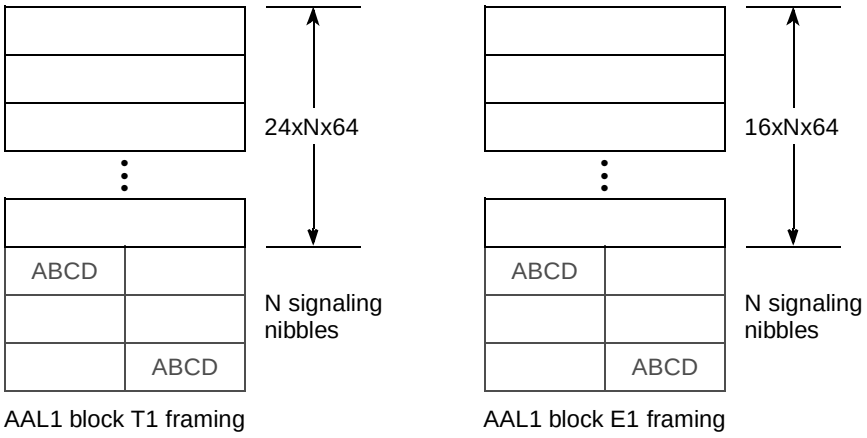
The QUICC Engine block supports synchronous residual time stamp (SRTS) generation using an external PLL. If this mode is enabled, the QUICC Engine block reads the SRTS code from external logic and inserts it into four outgoing cells. See Section 32.2.22.4.1, “SRTS Generation and Clock Recovery Using External Logic.”

35.2.2 Signaling Path

The QUICC Engine block automatically handles the signaling information as part of the interworking function. The ATM transmitter packs the signaling information at the end of each superframe during the data segmentation process. Each VC is associated to one signaling block by an internal routing table; see Section 35.4.6, “Channel Associated Signaling (CAS) Support.” The signaling information that resides in the internal RAM is inserted into the AAL1 cell according to the af-vtoa-0078 specification.

The AAL1 structure is divided into two sections. The first section carries the Nx64 payload, and the second carries the signaling bits that are associated with the payload.

The QUICC Engine block supports two framing modes: one for Nx64 DS1 ESF (extended superframe) framing, and the other for Nx64 E1 G.704 framing. See Figure 35-3. Each of the internal signaling blocks can be used to deliver only one of the framing formats; that is, they cannot be changed dynamically.



Note: The CAS block size is (N+1) nibble if N is an odd number.

Figure 35-3. AAL1 Framing Formats

35.3 AAL1 CES Receiver Overview

The ATM controller supports both AAL1 structured and unstructured formats. For the unstructured format, 47 octets are copied to the current receive buffer and for the structured format, 46 (P format) or 47 (non-P format) octets are copied to the current receive buffer.

The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is checked and the result is delivered to the 3-step-SN algorithm. The 3-step-SN algorithm (see Section 35.6.1, “The Three States of the Algorithm”) handles the lost or misinserted cells. This algorithm can detect one lost or misinserted cell and maintain synchronization. If more than one cell is lost or misinserted, the 3-step-SN algorithm switches to hunt mode where it stays until a cell with a valid SN field is received. After the receiver switches to hunt mode, it closes the RxBD, modifies the receive statistics, generates an optional interrupt to the CPU and performs a restart sequence.

The restart sequence is implemented only when the ATM channel works in CES mode (RCT[CESM]). In CAS mode (RCT[CASM]), the ATM receiver channel begins the restart sequence by dropping all incoming cells and advancing to the beginning of the next super frame, which is the first BD after the one marked with EOSF (end of super frame). When this BD is ready and the adaptive counter reaches the ATM_Start threshold, the receiver’s write pointer is no longer in danger of overrunning the read pointer of the MCC transmitter; that is, it is safe to begin receiving cells again. The ATM receiver then begins the resynchronization process: for unstructured AAL1 type the ATM receiver waits for the first valid cell, and for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame). (See Section 35.5, “ATM-to-TDM Adaptive Slip Control” and Section 35.4, “Interworking Functions.”)

Note that when the ATM channel is not in CES mode, no restart sequence is performed; the ATM receiver immediately starts hunting for the first valid cell. The first received octet becomes the first byte of the next BD.

During reassembly, the ATM receiver channel's 3-step-SN algorithm status is delivered to the pointer verification mechanism. (See Section 35.7, "Pointer Verification Mechanism.") If the receiver operates in unstructured data format, the 3-step-SN algorithm status is delivered directly to the bit count integrity module. When partially filled cells arrive, the bit count integrity module copies only the valid octets from the received cell (or from the dummy cell if that cell is lost) to the current receive buffer.

In unstructured AAL1 format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and begins receiving incoming cells.

In structured AAL1 format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer (valid pointers range between 0 and 93 inclusive, and the dummy pointer is 127) that points to the start of a new structure. When one arrives, the receiver leaves the hunt state, opens a new buffer and begins receiving. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer.

During the reassembly process, if the receiver switches to hunt mode (due to the 3-step-SN algorithm or due to receiving two successive mismatched pointers), the ATM receiver closes the current RxBD, discards incoming cells, modifies the receive statistics accordingly and initiates a restart sequence. The receiver then waits for a cell with a valid structured pointer to regain synchronization and start receiving incoming cells again.

The QUICC Engine block supports SRTS clock recovery using an external PLL (SRTS mode applicable only in unstructured mode). The QUICC Engine block tracks the SRTS from the four incoming cells and writes the SRTS code to external logic. See Section 32.2.22.4.1, "SRTS Generation and Clock Recovery Using External Logic." The data flow for an AAL1 CES receiver is summarized in Figure 35-4.

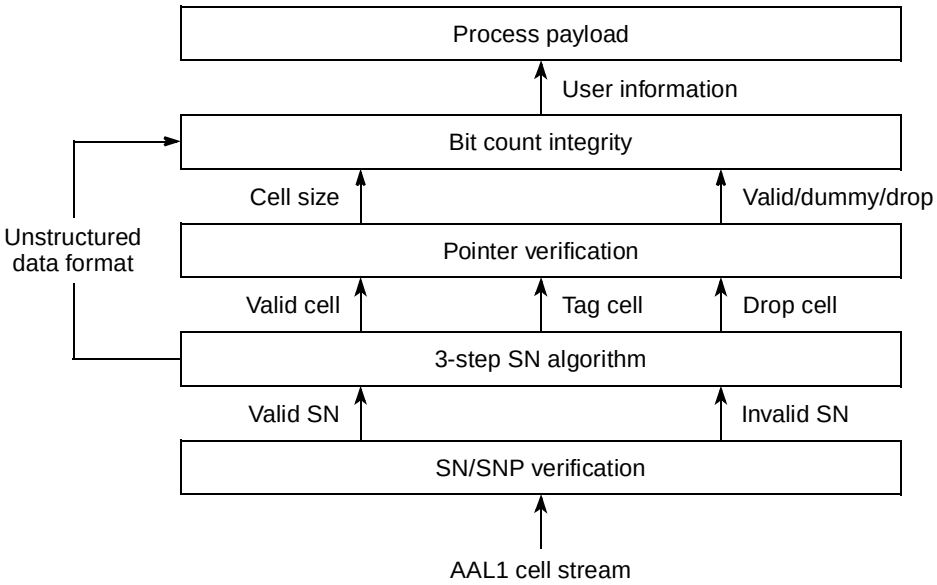


Figure 35-4. AAL1 CES Receiver Data flow

35.4 Interworking Functions

The QUICC Engine block supports the interworking of ATM and TDM. The TDM data processing is done by the MCC and the SI (refer to Chapter 34, "Multi-Channel Controller (MCC)," and Chapter 36, "Serial Interface with Time-Slot Assigner," for further information). The ATM controller processes the ATM data.

Data forwarding between the ATM controller and the MCC can be done in two ways:

- Automatic data forwarding. This mode enables automatic data forwarding between AAL1 and transparent mode over a TDM.
- Core intervention. When an MCC receive buffer is full and its RxBD is closed, the MCC interrupts the core. The core copies the MCC's receive buffer pointer to an ATM TxBD and sets the ready bit (TxBD[R]). Similarly, when an ATM receive buffer is full and its RxBD is closed, the core services the ATM controller's interrupt by copying the ATM receive buffer pointer to an MCC TxBD and setting TxBD[R]. This mode is useful when additional core processing is required.

Possible interworking applications are as follows:

- Circuit emulation service (CES)
- Carrying voice over ATM
- Multiplexing low speed services, such as voice and data, onto one ATM connection

35.4.1 Automatic Data Forwarding

The basic concept of automatic data forwarding is to program the ATM controller and the MCC to process the same BD table (ring).

The MCC and ATM receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive into buffers in which RxBD[E] = 0 and then set RxBD[E] when the buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1 CES-specific areas of the receive connection table; see Section 35.9.1, "Receive Connection Table (RCT)." For the MCC receiver, set CHAMR[EP].

35.4.1.1 ATM-to-TDM

When going from ATM to TDM (depicted in Figure 35-5), the ATM receiver reassembles data received from a particular channel to a specific BD table. The MCC transmitter is programmed to operate on the same table. When the ATM controller fills a receive buffer, the MCC controller sends it. The controllers synchronize on the ATM controller's RxBD[E] and the MCC's TxBD[R].

A threshold mechanism for the MCC transmitter is used to synchronize the automatic start of the ATM-to-TDM data forwarding. The transmitter waits until the start threshold is reached (enough cells are received) before sending the valid data. In effect, the MCC start threshold mechanism implements a jitter buffer designed to accommodate the CDV of the ATM network.

In a CES application, software should first initialize the MCC transmitter, then initialize the ATM receiver. This way the MCC transmitter sends idle data (0xFF) until the ATM receiver fills enough buffers to reach the MCC start threshold. (The receiver is effectively filling a jitter buffer.) When the CES_Adaptive_Counter (CESAC) reaches the MCC_Start threshold, the MCC super channel polling CESAC starts sending the received data with the first frame SYNC. (The first octet using the first BD is

sent on the first assigned time slot.) Section 35.5, "ATM-to-TDM Adaptive Slip Control," shows the flow of ATM-to-TDM interworking.

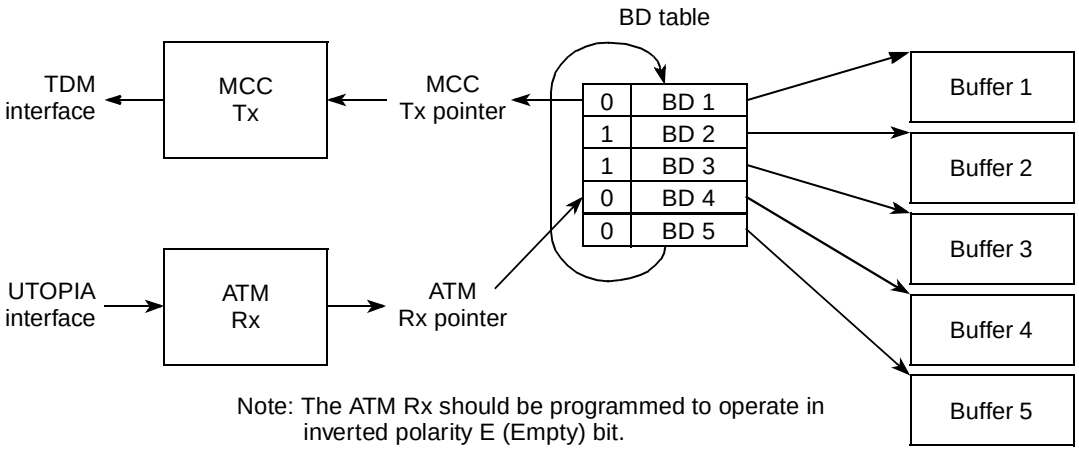


Figure 35-5. ATM-to-TDM Interworking

35.4.1.2 TDM-to-ATM

When going from TDM to ATM (depicted in Figure 35-6), the MCC receiver routes data from the TDM line to a specific BD table. The ATM transmitter is programmed to operate on the same BD table. When the MCC fills a receive buffer, the ATM controller sends it. The two controllers synchronize on the MCC's RxBD[E] and the ATM controller's TxBD[R].

In a CES application, first initialize the ATM transmitter, then initialize the MCC receiver and set CHAMR[EP].

In order to prevent an overrun condition on the MCC receiver, the ATM transmitter should be programmed to work at a faster rate than the MCC super channel. This ensures that the ATM channel polls the common BD table at a higher rate than it is being filled by the MCC. In CES mode, the ATM controller ignores BSY (busy) events when the ATM tries to open a buffer that is not yet full; it continues polling the BD until the buffer is filled by the MCC receiver and then updates the relevant statistics; see Section 35.15, "Internal AAL1 CES Statistics Tables."

Note that if an overrun condition occurs on the MCC, despite the above mechanism, software should restart the MCC and ATM channels in order to recover.

Figure 35-6 shows the flow of TDM-to-ATM interworking.

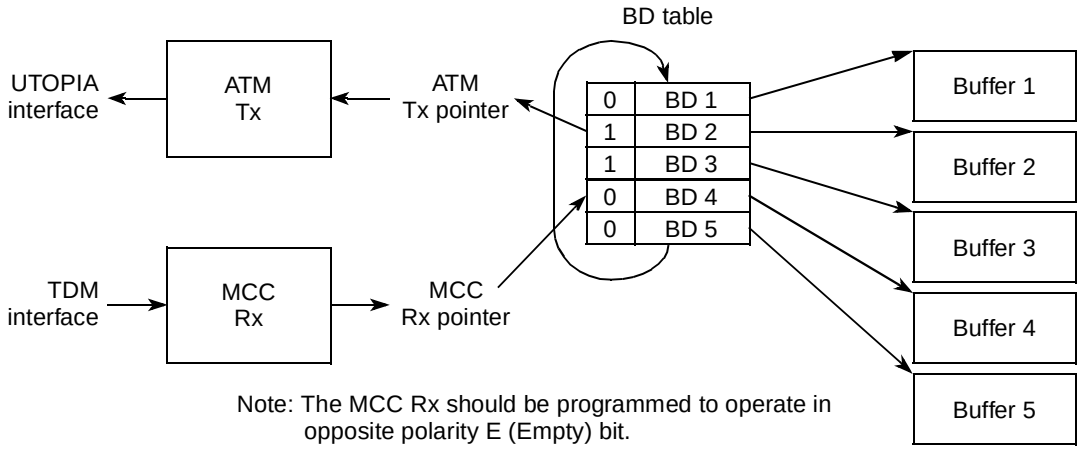


Figure 35-6. TDM-to-ATM Interworking

35.4.2 Timing Issues

Use of the TDM interface assumes that all communicating entities are synchronized; that is, that they are using a synchronized serial clock. If the TDM interfaces are not synchronized, a slip can occur in the reassembly buffer. In order to prevent the overrun and underrun condition, the QUICC Engine block maintains an adaptive slip control using a set of 4 threshold pointers and a counter for each ATM-TDM (VC to super channel) connection.

Before a buffer-not-ready event (ATM-to-TDM data forwarding) occurs at the MCC transmitter, the MCC buffer pointer reaches the MCC_Stop threshold. Consequently, the MCC pointer freezes on the last transmitted BD and starts sending the underrun template (or the last transmitted frame). In the meantime, the ATM receiver continues to write valid data and advance the ATM buffer pointer. When the adaptive counter CESAC reaches the MCC_start threshold and the MCC has finished sending a multiple frame size, the MCC exits the pre-underrun state, advances BD pointer, reads the new BD pointer and starts sending the valid received data. (Refer to Section 35.5, “ATM-to-TDM Adaptive Slip Control.”)

The same mechanism is implemented on the ATM side. When the ATM receiver (ATM-to-TDM data forwarding) reaches the ATM_Stop threshold (pre-overrun), the ATM controller switches to hunt mode and discards the channel’s incoming cells. In the meantime, the MCC transmitter continues to send valid data and advance the MCC buffer pointer. When CESAC falls to the ATM_start threshold, the ATM write pointer is advanced to the first BD after the one marked with EOSF (in CAS mode). When this BD is ready and CESAC reaches the ATM_Start threshold, the receiver’s write pointer is not longer in danger of overrunning the read pointer of the MCC transmitter; that is, it is safe to begin receiving cells again. The ATM receiver then begins the resynchronization process: for unstructured AAL1 type the ATM receiver waits for the first valid cell, and for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame). (Refer to Section 35.5, “ATM-to-TDM Adaptive Slip Control.”)

Note that when the ATM receiver is in hunt mode due to one of the following:

- Sequence number protection error (SNPE)

- Sequence count error (SCE)
- Structured pointer error (SPE)
- Slip condition

The signaling information (CAS) and SRTS information is not updated by the ATM controller until the ATM receiver switches to SYNC mode, that is, a valid cell is received in unstructured cell format or a valid pointer is received in structured cell formats.

Software should distinguish between the two types of overrun and underrun conditions:

7. The MCC and ATM controller can automatically recover from overruns and underruns caused by slips without any CPU intervention. (See Section 35.5, “ATM-to-TDM Adaptive Slip Control.”)
8. Global underrun (MCCE[GUN]) and overrun (MCCE[GOV]) conditions are errors that need CPU intervention because it is not known which channels are affected. The CPU should accordingly reinitialize the transmit parameters and/or the receive parameters to recover.

35.4.3 Clock Synchronization (SRTS, Adaptive FIFO)

Clock synchronization methods, such as SRTS and adaptive FIFO, may be used to prevent reassembly buffer slip. The SRTS method may be implemented using external logic. The QUICC Engine block can read the SRTS from external logic and insert it into outgoing AAL1 cells and conversely, can track the SRTS from incoming AAL1 cells and deliver it to external logic. See Section 32.2.22.4.1, “SRTS Generation and Clock Recovery Using External Logic.”

Alternatively, an adaptive FIFO method can be implemented under core control. Adaptive FIFO is a way to hold the bridging buffer at its mid-level point. One way to implement it is to periodically poll the adaptive counter CESAC (difference between the MCC and ATM data pointers) and use this difference as a pseudo-SRTS; see Section 35.5.1, “CES Adaptive Threshold Tables.” Writing the pseudo-SRTS to the same external PLL logic used in the SRTS method adjusts the TDM clock.

35.4.4 Mapping TDM Time Slots to VCs

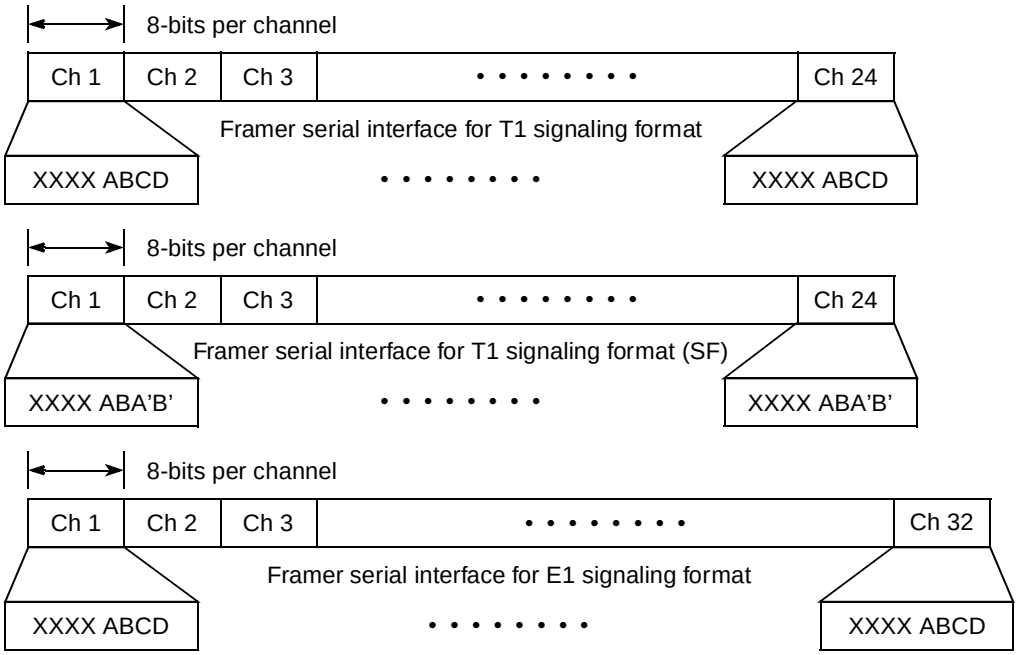
Any TDM time-slot combination can be routed to a specific data buffer using the MCC and its SI. (Refer to Chapter 34, “Multi-Channel Controller (MCC),” and Chapter 36, “Serial Interface with Time-Slot Assigner,” for further information.) A common set of data buffers (one BD table) should be used by the ATM controller to route both the receive and transmit data. For information about ATM buffers see Section 35.11, “Buffer Descriptors.”

35.4.5 Trunk Condition

According to the Bellcore standard, the interworking function should be able to transmit special payloads on both the ATM and TDM channels to signal alarm conditions (bellcore TR-NWT-000170). The trunk condition can be generated under core control. The core may deliver buffers containing special data (trunk condition payload) to the ATM controller or MCC or even overwrite data buffers being used by the channels.

35.4.6 Channel Associated Signaling (CAS) Support

For applications requiring channel associated signaling (CAS) support, the CAS manipulation is done by an external framer. The MCC should be programmed to receive or transmit the internal CAS block transparently through the external framer's serial interface. The internal CAS block (depicted in Figure 35-8) can be adjusted to comply with a specific framer's serial interface. (See Section 35.4.7.1, "CAS Routing Table.")



Note: The MCC should capture the signaling data on the last frame of a super frame. See Section 1.4.7.2, "TDM-to-ATM CAS Support".

Figure 35-7. Mapping CAS Data on a Serial Interface

The MCC and ATM CAS are synchronized with the superframe block boundary. At the ATM side, the structured block size should be set to the superframe block size plus the size of the CAS block so that the structured pointer inserted by the ATM controller points to the start of the structured data block. At the MCC side, the MCC is synchronized with the frame sync signal; the external framer has the ability to place the signaling information at the appropriate place in a superframe. The QUICC Engine block supports an automatic mode for forwarding the signaling information from the TDM to the ATM and vice versa. The ATM controller maintains two CAS blocks per trunk (eight total). One contains the signaling information unpacked from the AAL1 cells (ATM-to-TDM), and the other contains the signaling information fetched from an external framer (TDM-to-ATM). All 8 CAS blocks reside in the internal RAM.

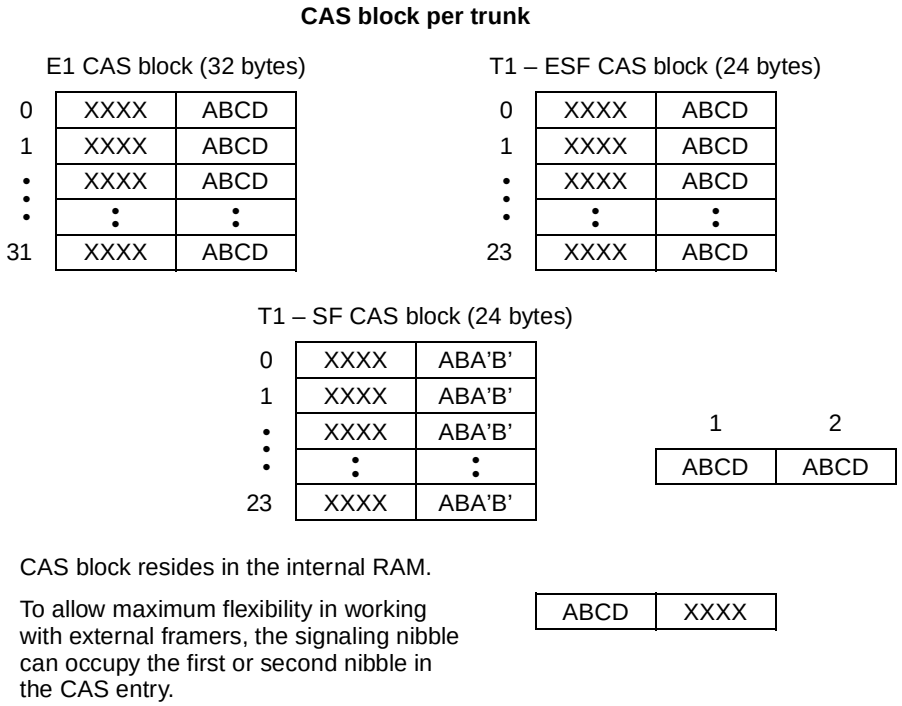
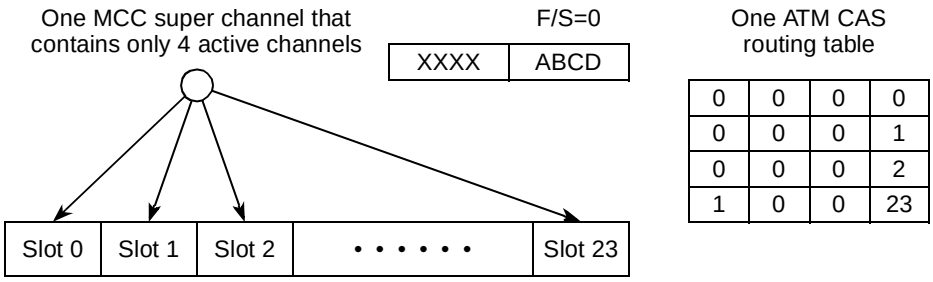


Figure 35-8. Internal CAS Block Formats

35.4.7 Mapping VC Signaling to CAS Blocks

Each ATM channel is connected to a specific CAS block. The ATM controller implements CAS routing tables (CRT) to maintain the routing of the signaling information from the AAL1 cells to the internal CAS blocks for receiving, and vice versa for transmitting. Each ATM channel has a routing table for the transmitter and one for the receiver. These tables are adjacent, each consists of a 32-byte space. Total size for both tables is 64 bytes with each entry pointing to one signaling nibble. To allow maximum flexibility with external framers, the signaling nibble can occupy the first or second nibble in the internal CAS block (depicted in Figure 35-8).

The CRT entries should be initialized only once before the ATM channel is enabled (receiver or transmitter). The number of entries that should be initialized must be equal to the number of active slots (channels) in the corresponding MCC super channel. Each super channel is mapped to a unique ATM connection (VC). The CRT entries are in ascending order based on the channel slots assigned for the MCC super channel (depicted in Figure 35-9).



Example of one MCC super channel in ESF framing (T1) containing 4 TDM slots connected to an ATM channel with one CAS routing table (CRT). See Section 1.4.7.1, “CAS Routing Table.”

Figure 35-9. Mapping CAS Entry

35.4.7.1 CAS Routing Table

Two CRTs are available per channel. The first is the transmitter table followed by a receiver table. Each table consists of 32 single byte entries. Accessing this table for the transmitter is done using: INT/EXT_RTCRT_BASE + CC * 64 and for a receiver INT/EXT+RTCRT_BASE + CC * 64 + 32.

Figure 35-10 shows the structure of a CAS routing table. The RISC maintains a pointer which steps through the table and wraps back to the beginning of the table after servicing the last entry (W=1).

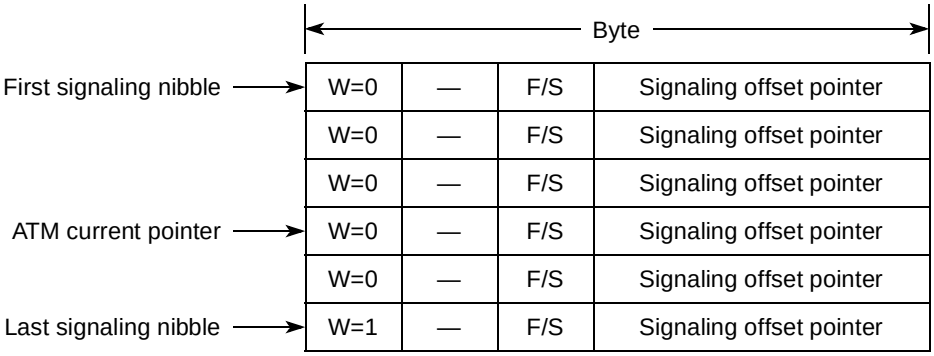


Figure 35-10. AAL1 CES CAS Routing Table (CRT)

Figure 35-11 describes the structure of a CAS routing table entry.

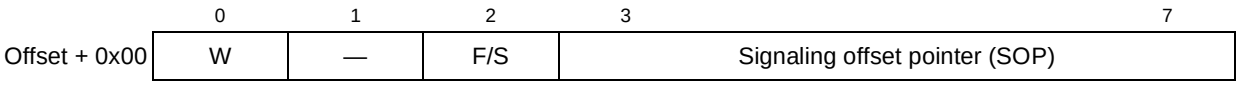


Figure 35-11. AAL1 CES CAS Routing Table Entry

Table 35-1 describes CAS routing table entry fields.

Table 35-1. CAS Routing Table Entry Field Descriptions

Bits	Name	Description
0	W	Wrap bit. When set, this bit indicates the last circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
1	—	Reserved, should be cleared during initialization.
2	F/S	First/Second. 0 Indicates that the signaling information occupies the **first nibble** in the CAS block (LSB). 1 Indicates that the signaling information occupies the **second nibble** in the CAS block (MSB).
3–7	SOP	Signaling Offset Pointer. Offset of the signaling nibble from the internal CAS base address. Note that in ESF mode the maximum offset is 23 and in E1 framing format the maximum offset is 31.

35.4.7.2 TDM-to-ATM CAS Support

During the segmentation process, the AAL1 CES transmitter reads the CAS data from the internal CAS block and packs the data and the signaling information at the end of an AAL1 super frame (depicted in Figure 35-3). All AAL1 functions operate normally (generating AAL1 PDU-headers, structured pointers, etc.). Each common (MCC, ATM) BD table should point to buffers that can contain a whole number of super frames. The last buffer of the super frame is marked as the end of a super frame (BD[EOSF]=1). After closing a buffer with the EOSF indication, the ATM transmitter processes the CAS data—reads it from the internal CAS block and inserts it into the cell payload at the transmit side. The EOSF indication in the BD is statically set by the CPU when initializing the BD table.

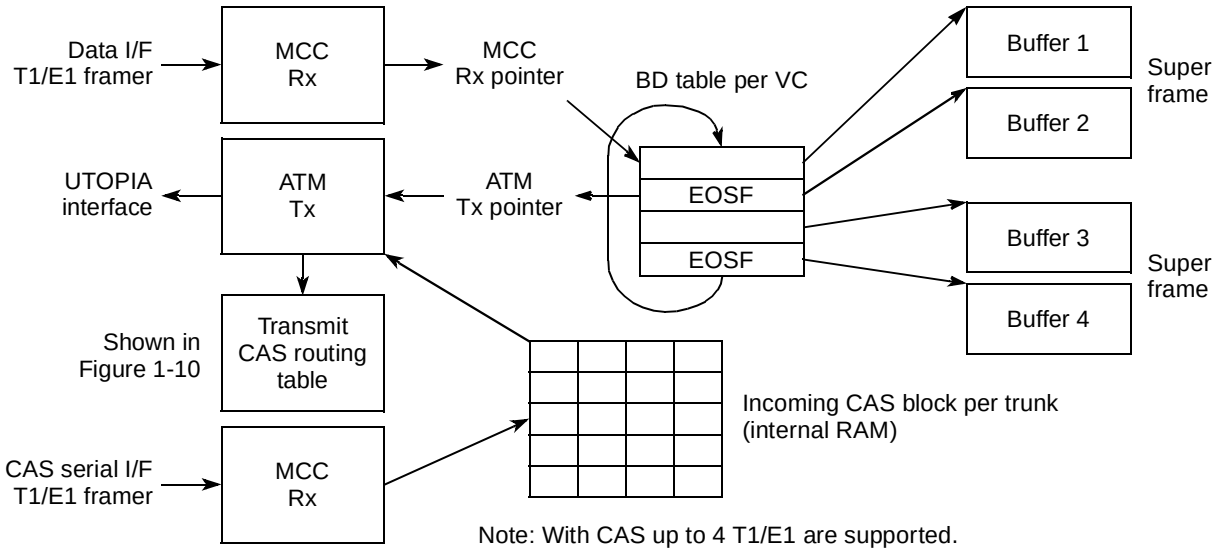


Figure 35-12. CAS Flow TDM-to-ATM

The CAS block is automatically written to internal RAM by the MCC receiver using a separate TDM. When a super frame is received the MCC should be triggered with a super-frame (multi-frame) SYNC from the external framer. The incoming CAS block should be captured by the MCC only once for each super frame (on the last frame).

The user may use external logic to convert the framer super-frame SYNC to trigger the MCC. The MCC captures the CAS block from the external framer and copies it transparently into one of four internal CAS blocks. Each byte in the CAS block contains a nibble of valid CAS information (depicted in Figure 35-8).

Note that the buffer data size should not include the CAS octets.

35.4.7.2.1 CAS Mapping Using the Core (Optional)

To avoid using another TDM dedicated to CAS information, the user can use a parallel interface (controlled by the core) to deliver the CAS information from the framer to the incoming CAS block. In this case, the core service routine reads the CAS information from the external framer and writes it to the incoming CAS block. To optimize the process, the framer can interrupt the core only when new information is received.

35.4.7.3 ATM-to-TDM CAS Support

During the reassembly process, the AAL1 CES receiver unpacks the signaling information from the end of an AAL1 super frame (depicted in Figure 35-3) and places it in the internal CAS block (using the receive CAS routing table). All AAL1 functions operate normally (AAL1 PDU-header verification, bit count integrity, 3-step-SN algorithm, etc.). Each common (MCC, ATM) BD table should point to buffers that can contain a whole number of super

frames. The last buffer of the super frame is marked with EOSF. After closing a buffer with an EOSF indication, the ATM receiver processes the CAS data (copies it to the internal CAS block from the AAL1 cell payload at the receive side). The EOSF indication in the BD is statically set by the CPU while initializing the BD table. See Figure 35-13.

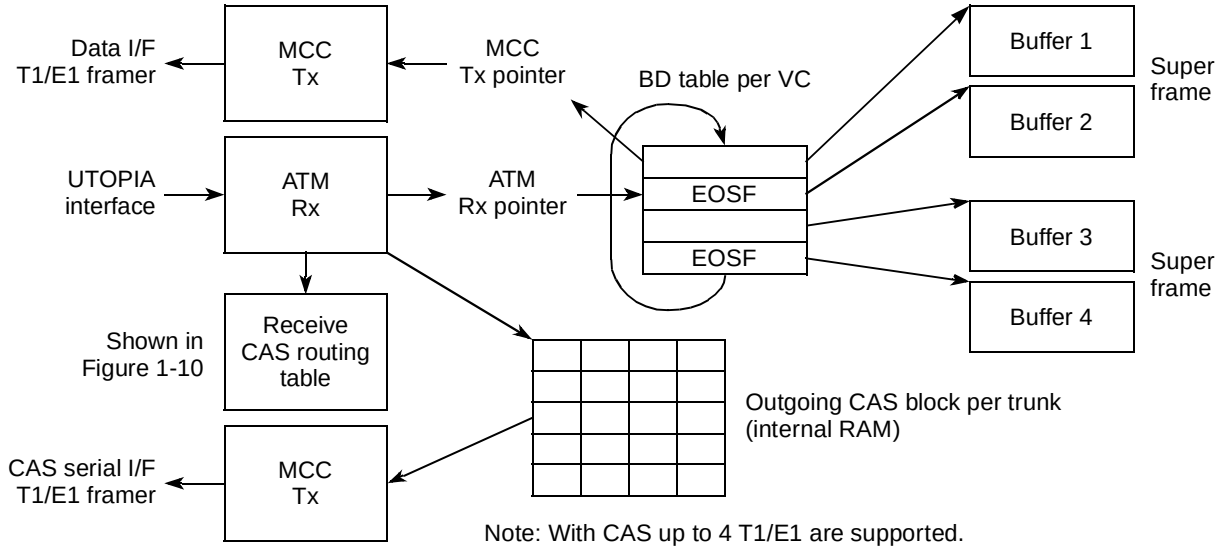


Figure 35-13. CAS Flow ATM-to-TDM

The CAS block is read from the internal RAM by the MCC. The MCC transmitter continuously reads the signaling information from one of the four outgoing internal CAS blocks and writes it transparently into

the external framer. Each byte in the CAS block contains one nibble of valid CAS information (depicted in Figure 35-8).

Note that the buffer data size should not include the CAS octets.

35.4.7.3.1 CAS Updates Using the Core (Optional)

To avoid using another TDM dedicated to CAS information, the user can use a parallel interface controlled by the core to deliver the outgoing CAS block to the framer. In this case, the ATM receiver should be programmed to operate in core CAS modify mode RCT[CCASM=1] (and RCT[CASM=1]). In this mode, the RISC adds an entry to the ATM interrupt queue and sets the appropriate sticky bit in OCASSR each time an AAL1 cell is received with new signaling information (one or more signaling nibble has changed). (See Section 35.10, "Outgoing CAS Status Register (OCASSR).") A core interrupt service routine can then write the updated outgoing CAS block to the framer.

Note to avoid additional latency, the interrupt queue assigned to this connection should have a global interrupt threshold of one. See the INT_ICNT parameter discussed in Section , ".

35.5 ATM-to-TDM Adaptive Slip Control

Two types of slip can occur in ATM-to-TDM operation: overrun and underrun. The two cases are handled by the MCC and ATM controller automatically without requiring CPU intervention.

Overrun occurs when the MCC transmitter fetches data from the common BD table at a slower rate than it is being filled by the ATM receiver. In this case, the ATM write pointer meets the MCC read pointer and a BSY state is declared (an entry is added to the ATM interrupt table) on the ATM side.

Underrun occurs when the MCC transmitter fetches data from the common BD table at a higher rate than it is being filled by the ATM receiver. In this case, the MCC read pointer reaches a BD that is not ready and a buffer-not-ready state is declared on the MCC side.

In both slip cases, the MCC and ATM controller automatically recover and restart the ATM-to-TDM interworking function.

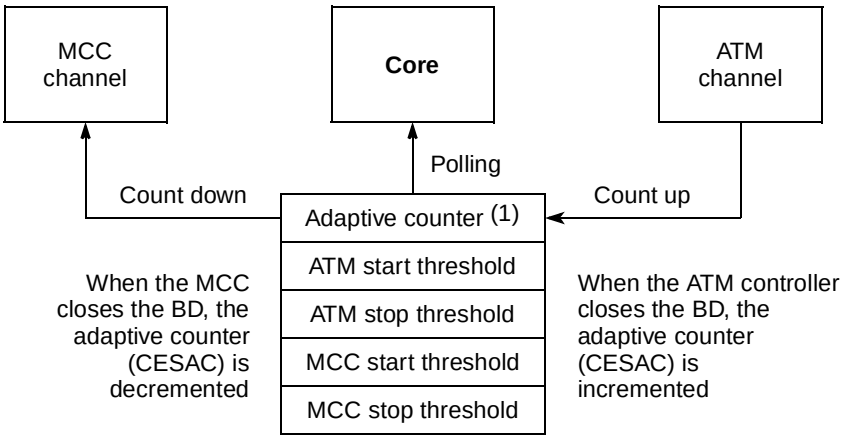
In order to prevent overrun and underrun conditions, the QUICC Engine block maintains an adaptive slip control using a set of 4 threshold pointers for each ATM-TDM (VC - super channel) connection.

The pre-underrun state (shown in Figure 35-14) occurs when the MCC read pointer goes faster than the ATM write pointer. When the adaptive counter reaches the MCC_Stop threshold, the MCC read pointer does not advance. At this point, the current BD (or the underrun template) is retransmitted a multiple of the frame size. In the meantime, the ATM receiver continues to receive valid data and advance the ATM write pointer. When the adaptive counter reaches the MCC_start threshold and the MCC has finished sending a multiple of the frame size, the MCC starts to transmit the valid received data and advance the MCC read pointer.

Note that when the pre-underrun state occurs, the MCC transmitter can transmit the last buffer continuously or the underrun template. This is determined by the MCC channel configuration; see the CHAMR[UTM] bit description in Section 34.2.2.2.3, "Channel Mode Register (CHAMR)—Transparent Mode." In the example shown in Figure 35-14, the MCC is programmed to send the current BD during the pre-underrun condition.

The pre-overrun state occurs when the ATM write pointer goes faster than the MCC read pointer. When the adaptive counter reaches the ATM_Stop threshold, the ATM write pointer does not advance. The ATM receiver waits until the adaptive counter reaches the ATM_start threshold. In the meantime, the MCC read pointer continues to process valid data from the common BD table. When CESAC reaches the ATM start threshold, the ATM write pointer advances to the first BD after the one that marked with EOSF (in CAS mode). When this BD is ready, the ATM receiver begins the resynchronization process: for unstructured AAL1 type the ATM receiver waits for the first valid cell, and for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame).

Note that this implementation for slip control provides a good interface for an adaptive FIFO implemented in software. CESAC represents the difference between the ATM and MCC pointers; the software application need only convert this value into an SRTS format. Figure 35-14 shows the 8-byte data structure used to implement ATM-to-TDM slip control. (Three of the bytes are unused.)



CES adaptive threshold table address: CATB + MCC_Super_Channel*8

Notes:

1. The MCC start threshold, in effect, implements a CDV jitter buffer.
2. MCC stop threshold should be programmed to (BD Table size -b) to prevent buffer underrun.
3. ATM stop threshold should be programmed to (BD Table size - a) to prevent buffer overrun.
4. The MCC and ATM stop thresholds determine the CDVT.
5. The ATM start threshold determines the time the ATM receiver waits before restarting synchronization process.
6. (MCC start - MCC stop) >= frame size.
7. b and a are integers less the BD table size.

Figure 35-14. Data Structure for ATM-to-TDM Adaptive Slip Control

35.5.1 CES Adaptive Threshold Tables

The CES adaptive threshold tables (see Table 35-15) reside in the multi-user RAM and hold the CES thresholds on a per-VC basis. The CES adaptive threshold base (CATB), located in the AAL1 CES parameter RAM, points to the base address of these tables. Each AAL1-MCC channel has its own table with a starting address given by CATB + RCT[Super_Channel_Number]# × 8.

	0 — 7	8 — 15
Offset + 0x00	—	CES Adaptive Counter
Offset + 0x02	ATM Stop Threshold	ATM Start Threshold
Offset + 0x04	MCC Stop Threshold	MCC Start Threshold
Offset + 0x06	—	

Figure 35-15. CES Adaptive Threshold Table

Table 35-2 describes CES adaptive threshold table fields.

Table 35-2. CES Adaptive Threshold Table Field Descriptions

Offset[1]	Bits	Name	Description
0x00	0-7	—	Reserved, should be cleared during initialization.
	8-15	CESAC	CES adaptive counter.This field contains the difference between the ATM write pointer and the MCC read pointer on the common BD table.This field should be cleared by the user during the channel initialization.
0x02	0-7	ASTP	ATM stop threshold.This threshold determines the maximum amount of buffering (CDVT) that required in the common BD table (shown in Figure 35-16). When the CESAC reaches this value a pre-overrun condition occurs.This field should be defined by the user during the channel initialization.
	8-15	ASTRT	ATM start threshold.This threshold determines the Time interval that will take the ATM controller to restart the synchronize process (shown in Figure 35-16) after pre-overrun condition.This field should be defined by the user during the channel initialization.
0x04	0-7	MSTP	MCC stop threshold.This threshold determines the minimum amount of buffering in the common BD table (shown in Figure 35-16).When the CESAC reaches this value a pre-underrun condition occur and the MCC starts to transmit the underrun template or the last BD (see Section 35.5, “ATM-to-TDM Adaptive Slip Control”).This field should be defined by the user during the channel initialization.
	8-15	MSTRT	MCC start threshold.This threshold determines the effective depth of the jitter buffer, the amount of buffering required to cope with the ATM network’s CDV (shown in Figure 35-16).When the CESAC reaches this value the MCC transmitter starts sending the valid data from the common BD table.This field should be defined by the user during the channel initialization.
0x06	0-15	—	Reserved, should be cleared during initialization.

[1] The offset is CATB + RCT[Super_Channel_Number]# × 8

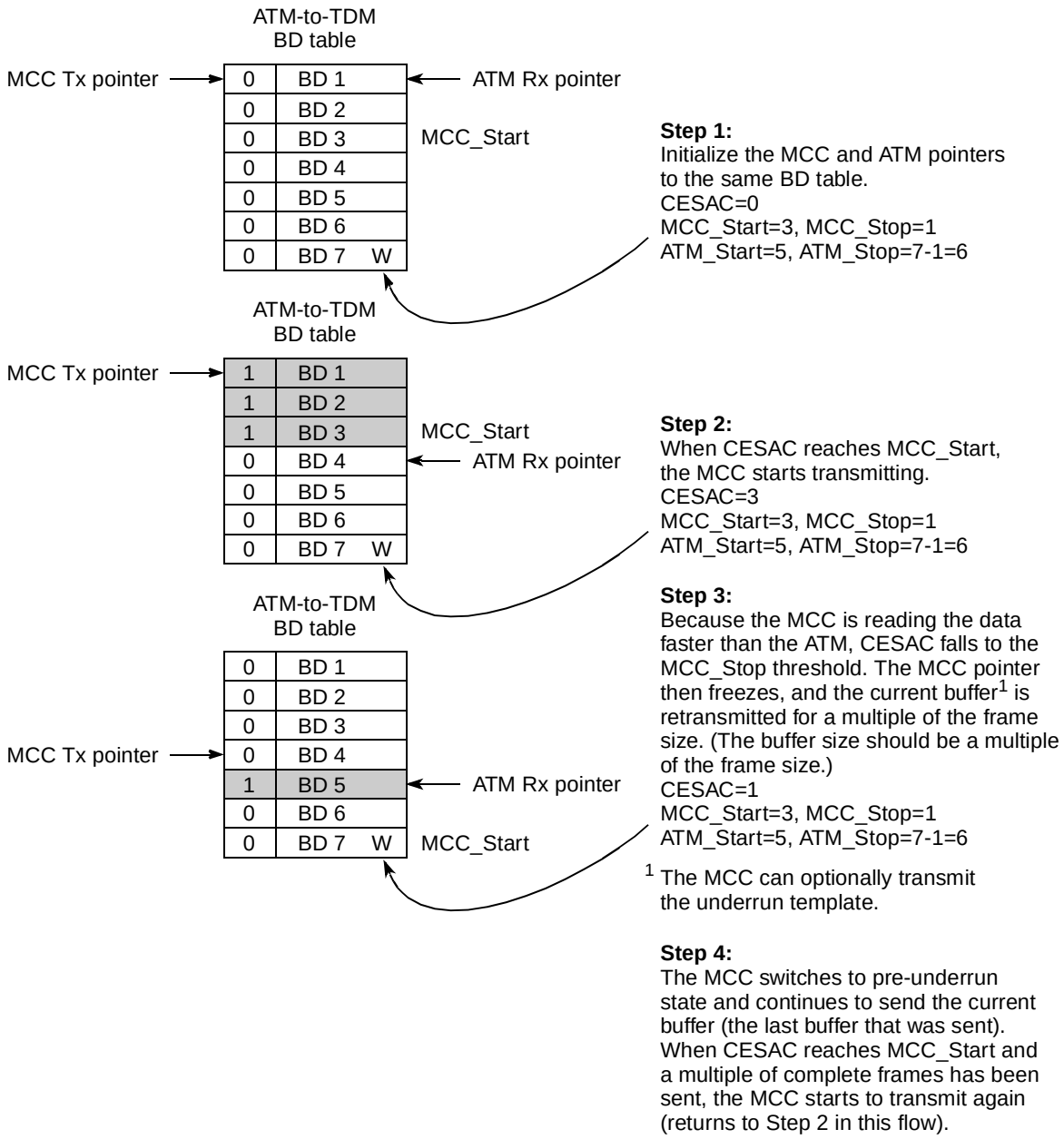


Figure 35-16. Pre-Underrun Sequence

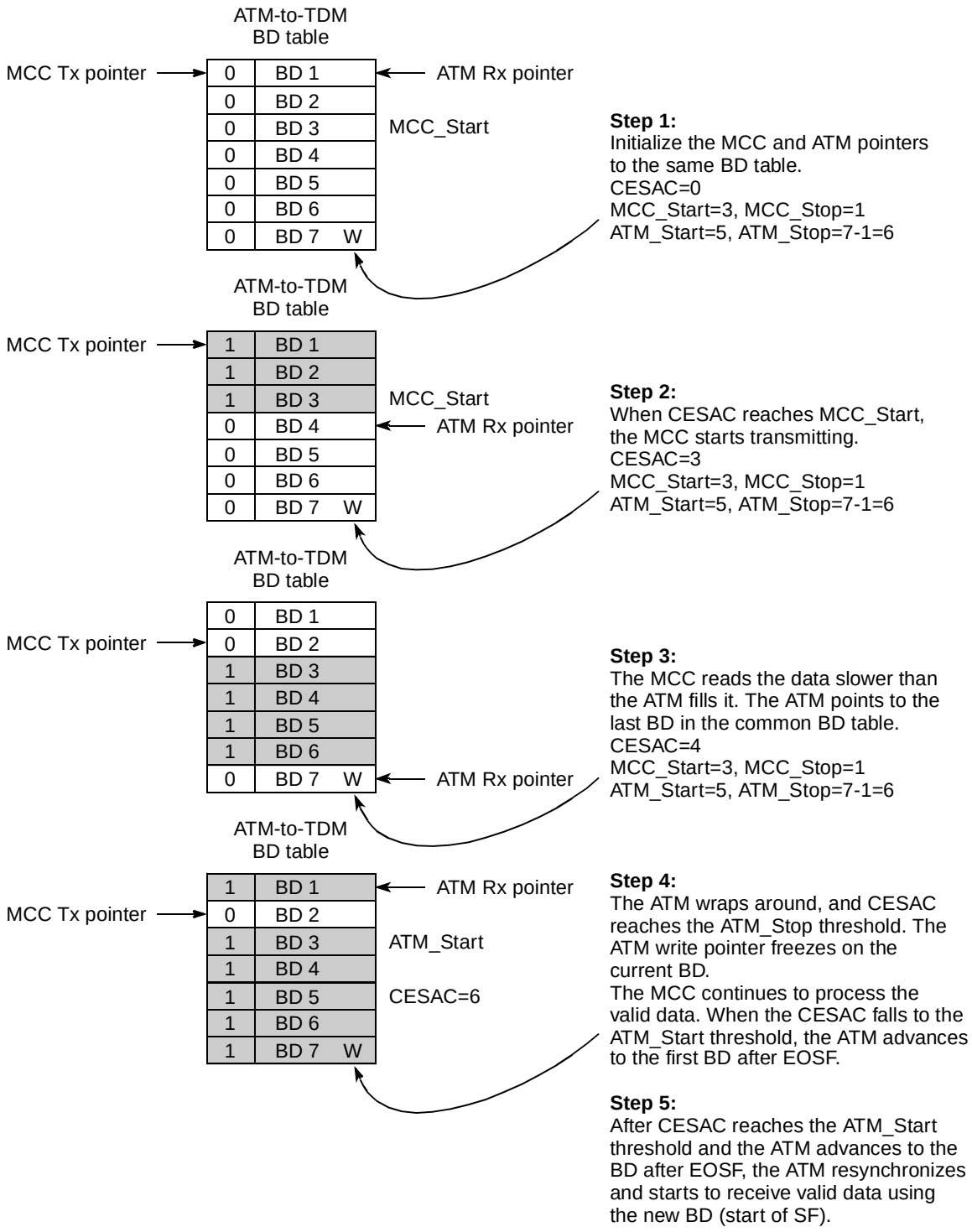


Figure 35-17. Pre-Overrun Sequence

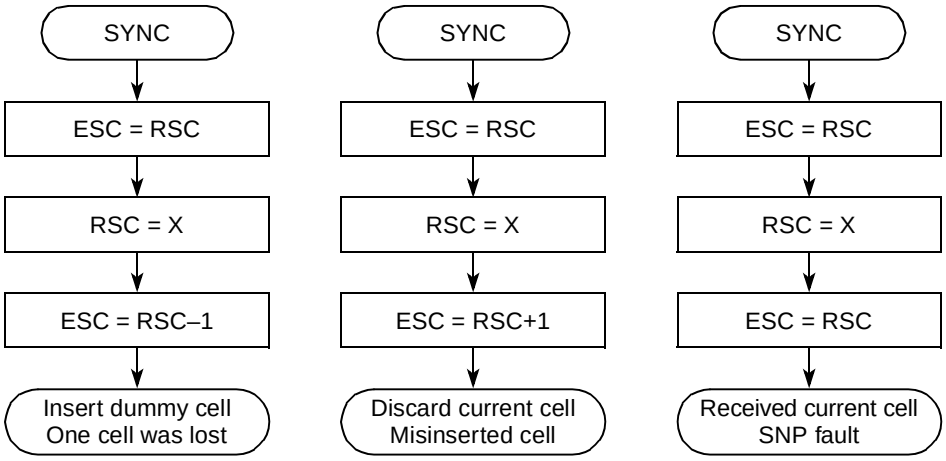
35.6 3-Step-SN Algorithm

The 3-step-SN algorithm is a fast and efficient state machine that has the ability to recover one lost or misinserted cell. The 3-step-SN algorithm does not add significant delay to the AAL1 CES reassembly process due to the fact that the decision to accept a cell is taken immediately after cell arrival. If a lost cell is detected, the receiver will insert a dummy cell. If a misinserted cell is detected, the receiver will discard the cell received after the misinserted cell. If more than one successive cell was lost or misinserted, the 3-step-SN algorithm will switch to hunt mode, where it stays until a cell with a valid SN field is received.

35.6.1 The Three States of the Algorithm

The 3-step-SN algorithm has three states:

1. Hunt—The Hunt state is the initial state of the algorithm or the first state after the receiver loses its synchronization. In this state no cell is delivered to the Rx buffer. When a valid cell is detected, the algorithm switches to the Sync state and delivers the current cell to the Rx buffer.
2. Sync—The Sync state is the steady state of the algorithm. In this state any received cell is automatically passed to the Rx buffer. The algorithm will switch from this state when an invalid cell is received (SNP error), or it receives a cell that does not sequence with the last valid cell (SC error).
3. Sync Fail—The Sync Fail state is a transient state. In this state the receiver checks the difference between the received sequence count (RSC) and the expected sequence count (ESC). If the difference between the two (ESC-RSC) is -1, 0, or 1, the receiver switches to sync mode and either discards the current cell, receives the current cell, or inserts a dummy cell, correspondingly. In any other cases the receiver will switch to hunt mode and discard the current cell.



ESC: Expected sequence count.
RSC: Received sequence count.
RSN=X: Invalid cell received or, a cell that does not match to the ESC.

Figure 35-18. Recoverable Sync Fail Sequence Options

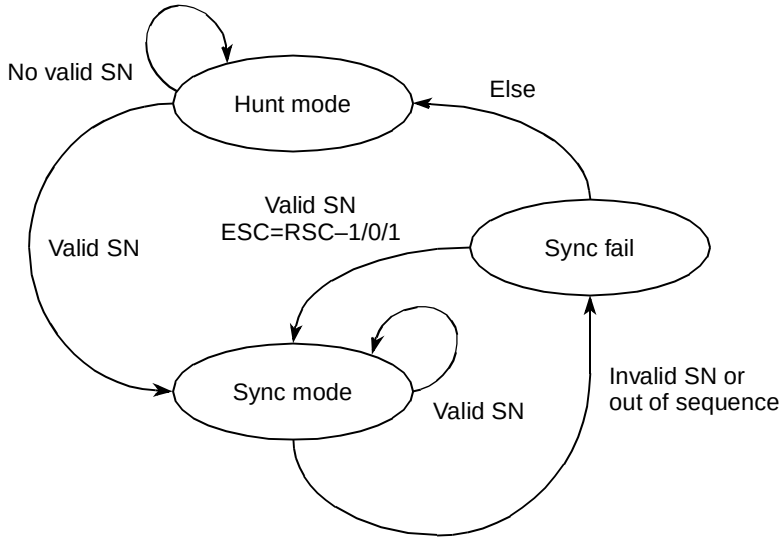


Figure 35-19. 3-Step-SN-Algorithm

35.7 Pointer Verification Mechanism

After the 3-step-SN algorithm processes the incoming cells, the cells status (Valid, Tag, Drop or Dummy) is delivered to the pointer verification mechanism. This state machine calculates the expected received pointer. If the current cell is valid and supposed to deliver a pointer, the received pointer is compared with the expected one. In the case of pointer mismatch or pointer error (i.e parity error), the pointer state machine switches to “Pre-Hunt-Mode”. If the cell status is not valid (i.e Tag, Drop or Dummy) and it supposed to carry a pointer, the pointer declared a mismatch and the pointer state machine switches to “Pre-Hunt-Mode”. The receiver stays in this transient state for only one AAL1 cell cycle (8 successive cells). When the pointer received in this state is different from the expected one or had a non-valid status, the receiver switches to hunt mode where it stays until a valid cell with a “start” structured pointer is received to regain synchronization.

Note that a “start” pointer is a valid pointer with a value not equal to 93or 127.

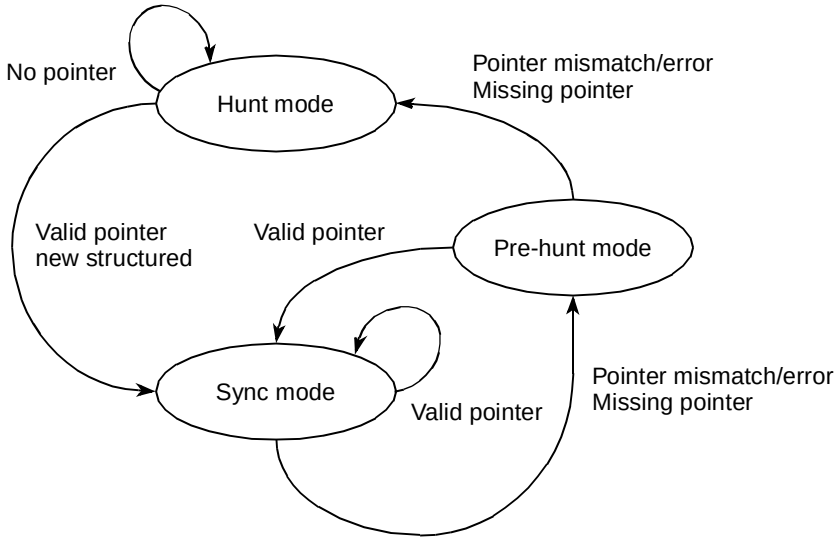


Figure 35-20. Pointer Verification Mechanism

35.8 AAL-1 Memory Structure

The QUICC Engine block manages ATM traffic by means of transmit and receive buffer descriptors (BD) and by transmit and receive connection tables (referred to as TCTs and RCTs, respectively). Buffer descriptors are circular lists of pointers into transmit and receive buffer space in external memory. The following sections describe the organization and configuration of the buffer descriptors, TCTs, and RCTs.

35.8.1 AAL1 CES Parameter RAM

The QUICC Engine parameter RAM is used to configure the three FCCs. The RISC also uses the parameter RAM to store operational and temporary values. When configured for ATM mode, the QUICC Engine block overlays the configuration structures shown in the next tables. The following table includes fields for AAL1 CES.

Note that some of the values must be initialized by the user; values set by the RISC should not be modified by the user.

Table 35-3. AAL1 CES Parameters

Location: Table 32-18	Name	Width	Description
Offset 0x36	INT_RTCRT_BASE	Hword	Internal receive/transmit CAS routing table extension base. User-defined. Should be 64-byte aligned. Each transmitter entry is accessed using: INT_RTCRT_BASE + CC * 64 and for a receiver INT_RTCRT_BASE + CC * 64 + 32.
Offset 0x38	EXT_RTCRT_BASE	Word	External receive/transmit CAS routing table extension base. User-defined. Should be 64-byte aligned. Each transmitter entry is accessed using: EXT_RTCRT_BASE + CC * 64 and for a receiver EXT_RTCRT_BASE + CC * 64 + 32.
Offset 0x4A	AAL1_INT_RX_CRT	Hword	(CES only) Points to a reserved scratchpad area of 32 bytes in the multi-user RAM used by the CES microcode. Should be 32 byte aligned. User-defined.

Table 35-3. AAL1 CES Parameters (continued)

Offset 0x48	OCASSR (AAL1_Out_CAS_Status_Reg)	Byte	Outgoing CAS Status Register. See Section 35.10, "Outgoing CAS Status Register (OCASSR)."
Offset 0x28	TCELL_TMP_BASE_EXT	Word	Transmit Cell Temporary base address (64-byte aligned). Points to an external memory block reserved for partially filled cells (64 octets for each CES channel). This area is allocated by the user but used by the RISC.
Offset 0x44	IN_CAS_BLOCK_BASE	Hword	Incoming CAS Block Base (depicted in Figure 35-12). Points to multi-user RAM. Should be 32-byte aligned. User-defined.
Offset 0x46	OUT_CAS_BLOCK_BASE	Hword	Outgoing CAS Block Base (depicted in Figure 35-10). Points to multi-user RAM. Should be 32-byte aligned. User-defined.
Offset 0x4C	AAL1_Int_STATT_BASE (AAL1_Int_STATS_BASE)	Hword	AAL1 Internal Statistics Table Base. Points to multi-user RAM. Should be 16-byte aligned. User-defined. See Section 35.15, "Internal AAL1 CES Statistics Tables."
Offset 0x34	AAL1_DUMMY_CELL_BASE	Hword	ALL1 Dummy cell base address. Points to multi-user RAM area contains the AAL1 dummy cell template (little-endian format). Should be 64-byte aligned. User-defined.
Offset 0x3E	CATB (ADAPTIVE_THRESHOLDS_BASE)	Hword	CES adaptive threshold tables base address. Points to the multi-user RAM area containing the CES slip control thresholds and the adaptive counter See Section 35.15, "Internal AAL1 CES Statistics Tables." Should be 8-byte aligned (8 octets for each AAL1-MCC channel). User-defined and should match the CATB value programmed in the MCC parameter RAM; see Section 34.2.2.3, "MCC Parameters for AAL1 CES Usage."
Offset 0x30	AAL1_Ext_STATT_BASE (AAL1_Ext_STATS_BASE)	Word	AAL1 External Statistics Table Base. Points to External memory. Should be 16-byte aligned (16 octets for each AAL1 channel).User-defined. See Section 35.16, "External AAL1 CES Statistics Tables."

35.9 Receive and Transmit Connection Tables (RCT, TCT)

The connection tables, RCT and TCT, hold channel configuration and temporary parameters for each receive and transmit channel (AAL type, connection traffic parameters, BDs' parameters and temporary parameters used during segmentation and reassembly).

The internal connection tables hold parameters for up to 128 channels (channels 0–127). In extended channel mode, parameters for channels 256 and above are kept in external memory. The only relationship between transmit and receive connection tables of the same channel is the CRT (CAS routing table); see Section 35.4.7.1, "CAS Routing Table."

Each connection table entry resides in 32 bytes. The pointers to these connection tables reside in the parameter RAM.

35.9.1 Receive Connection Table (RCT)

Figure 35-21 shows the format of an RCT entry.

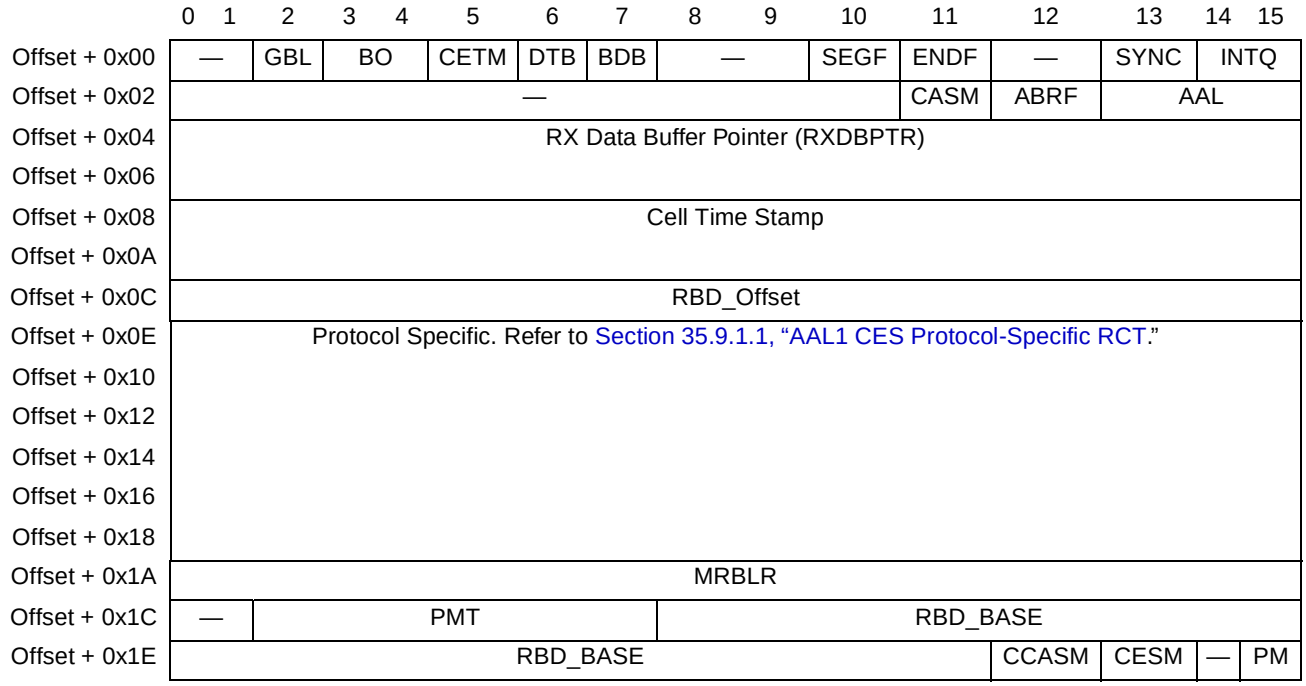


Figure 35-21. Receive Connection Table (RCT) Entry

NOTE

For an active channel, the RISC uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes.

Table 35-4 describes RCT fields.

Table 35-4. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	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 19.1.8.2, "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 M1BDBSCRD4 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 19-1 and Section 5.4.3.7, "Debug Configuration."
	6	DTB	Data buffers bus 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the QUICC Engine secondary bus.
	7	BDB	BD interrupt queues, free buffer pool, and CES external statistics tables bus placement 0 Reside on the coherent system bus. 1 Reside on the QUICC Engine secondary bus. Notes: 1. When in UDC mode (AAL5 or AAL1 or AAL1 CES), BDs and data should be placed on the same bus (RCT[DTB]=RCT[BDB]). 2. RCT[BDB] programming must be consistent across all CES receive channels because they share the same AAL1_Ext_STATT_BASE parameter.
	8–9	—	Reserved, should be cleared during initialization.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI=100 to the raw cell queue. 1 Send cells with PTI=100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12	—	Reserved, should be cleared during initialization.
	13	SYNC	AAL1 SYNC. The user should set this bit for first AAL1 synchronization. Used internally by the RISC.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 35-4. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-3	—	Internal use only. Initialize to 0.
	4–10	—	Internal use only. Initialize to 0.
	11	CASM	Common associated signaling mode. 0 CAS operation mode is disable. 1 CAS operation mode is enable.
	12	—	Reserved, should be cleared during initialization.
	13–15	AAL	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 010 AAL5 —ATM adaptation layer 5 100 AAL2 —ATM adaptation layer 2. Refer to Chapter 32, "ATM Controller AAL0, AAL1, and AAL5." 101 AAL1_CES —ATM adaptation layer 1 with circuit emulation service All others reserved.
0x04	—	RxDBPTR	Receive data buffer pointer. Holds real address of current position in the Rx buffer.
0x08	—	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 20.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)."
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel's current BD. User-initialized to 0; updated by the RISC.
0x0E–0x18		—	Protocol-specific area.
0x1A	—	MRBLR	Maximum receive buffer length. Used in both static and dynamic buffer allocation. For better synchronization between ATM and TDM, while CES traffic is steady, or when re-sync is needed, this value must be a multiple of the number of slots used by the TDM super channel associated with this AAL1 channel. When working in CAS mode this value must be the size of the data block, meaning each buffer hold one data block. The value of the data block is computed as follows: Let N be the number of slots used by the TDM super channel associated with this AAL1 channel. Then for a Super Frame: Data block size = 16*N. For an Extended Super Frame: Data block size = 24*N

Table 35-4. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. 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	CCASM	Core CAS modify. When this mode is enabled, the RISC sets OCASSR[MCASB*n*] sticky bit each time the outgoing (ATM to TDM) CAS block is changed. 0 Core CAS modify mode is disabled. 1 Core CAS modify mode is enabled. See Section 35.10, "Outgoing CAS Status Register (OCASSR)."
	13	CESM	Circuit Emulation Service Mode. 0 CES operation mode is disable. Adaptive Slip control mechanism is disabled. 1 CES operation mode is enable. Adaptive Slip control mechanism is enabled.
	14	—	Reserved, should be cleared during initialization.
	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.

35.9.1.1 AAL1 CES Protocol-Specific RCT

Figure 35-22 shows the AAL1 CES protocol-specific area of an RCT entry.

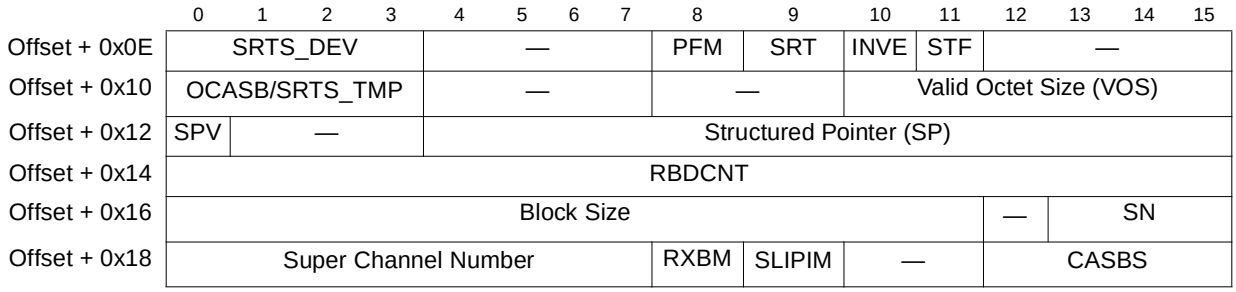


Figure 35-22. AAL1 CES Protocol-Specific RCT

Table 35-5 describes AAL1 CES protocol-specific RCT fields.

Table 35-5. AAL1 CES Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–3	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.
	4–7	—	Reserved, should be cleared during initialization.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The QUICC Engine block supports clock recovery using an external SRTS PLL. The QUICC Engine block tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the RISC writes a valid SRTS to external logic. (See Section 32.2.22.4.1, “SRTS Generation and Clock Recovery Using External Logic.”) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty). Note that in CES mode (CESM=1) this bit must be set by the user; see Section 35.4.1, “Automatic Data Forwarding.”
	11	STF	Structured format. 0 Unstructured format is used. 1 Structured format is used. Note that although the structured format may be used, if the block size is one, the user should clear STF. Only non P-format AAL1 cells are received. The receiver does not check the AAL1 structured pointer because there is no need to when the block size is one.
	12-15	—	Reserved, should be cleared during initialization.
0x10	0–3	OCASB/SRTS_TMP	OCASB applies when in CAS mode. Outgoing CAS Block. Points to one of the eight available internal CAS blocks. The starting address of the table is OUT_CAS_BLOCK_BASE+OCASB¥32. See Section 35.4.7.1, “CAS Routing Table” for more details. Note that the RCT and TCT use the same CAS routing table (CRT). SRTS_TMP applies when not in CAS mode. Used by the RISC to store the received SRTS code. After a cell with SN = 7 is received, the RISC writes the SRTS code to the external SRTS device. Note that when the receiver is in hunt mode SRTS information is not updated.
	4–9	—	Reserved, should be cleared during initialization.
	10–15	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only.
0x12	0	SPV	Structured pointer valid. Should be user-initialized user to zero. Structured format only.
	1–3	—	Reserved, should be cleared during initialization.
	4–15	SP	Structured pointer. Used by the RISC to calculate the structured pointer. This field should be initialized by the user to zero. Used in structured format only.

Table 35-5. AAL1 CES Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x14	—	RBDCNT	RxBD count. Indicates how many bytes remain in the current Rx buffer. Initialized with MRBLR whenever the RISC opens a new buffer.
0x16	0–11	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum). Note that when working in CAS mode (CESM=1 and CASM=1), the block size should be programmed to the superframe block size plus the size of the CAS block. However, when working in basic mode (CES without CAS: CESM=1 and CASM=0) the block size should be programmed to the number of MCC slots (Nx64) per frame assigned to this channel.
	12	—	Reserved, should be cleared during initialization.
	13–15	SN	Sequence number. Used by the RISC to check incoming cell's sequence number.
0x18	0-7	SCN	Super Channel Number. This field should contain the MCC Super Channel Number that mapped to this ATM channel. This field must be initialized by the user in CES mode only. See Section 35.4.7, "Mapping VC Signaling to CAS Blocks."
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is disabled. (The event is not sent to the interrupt queue.) 1 The receive buffer event of this channel is enabled.
	9	SLIPIM	Slip interrupt mask 0 The slip interrupts SLIPS and SLIPE are both masked. 1 When the receiver switches to hunt mode due to a 3-step-SN algorithm fault, or due to two successive mismatched pointers, or due to a pre-overrun condition, the SLIPS interrupt is sent to the interrupt queue. See Section 35.6, "3-Step-SN Algorithm," and Section 35.13, "AAL1 CES Exceptions." When the receiver resynchronizes, SLIPE interrupt is sent to the interrupt queue. Note that this is the error reporting mechanism during automatic data forwarding (ATM-to-TDM bridging).
	10-11	—	Reserved, should be cleared during initialization.
	12-15	CASBS	CAS block size.This field contains the number divided by two (**N/2**) of signaling nibbles mapped to this ATM channel. See Section 35.4.7, "Mapping VC Signaling to CAS Blocks." **Note** that if the number of signaling nibbles is **odd**, this field should be programmed to (**N+1)/2**. See Section 35.2.2, "Signaling Path." **Example**: ESF with three T1 time slots connected to one ATM channel. The **Data_Size** = 3*24 = 72 octets and the **CAS Block= 2** octets (**N=3, (N+1)/2 = 2**), so in this case the **Block_Size** = 72+2 = 74

35.9.2 Transmit Connection Table (TCT)

Figure 35-23 shows the format of an TCT entry.

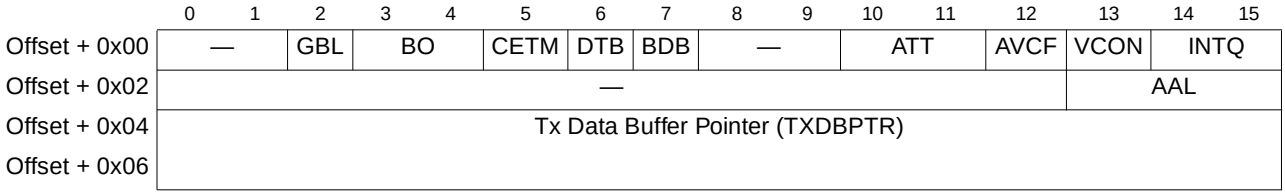


Figure 35-23. Transmit Connection Table (TCT) Entry

Offset + 0x08	TBDCNT						
Offset + 0x0A	TBD_OFFSET						
Offset + 0x0C	Rate Remainder		PCR Fraction				
Offset + 0x0E	PCR						
Offset + 0x10	Protocol Specific. Refer to Section 35.9.2.1, “AAL1 CES Protocol-Specific TCT.”						
Offset + 0x12							
Offset + 0x14							
Offset + 0x16	APC Linked Channel						
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)						
Offset + 0x1a							
Offset + 0x1C	—	PMT	TBD_BASE				
Offset + 0x1E	TBD_BASE			BNM	STPT	IMK	PM

Figure 35-23. Transmit Connection Table (TCT) Entry (continued)

Table 35-6 describes general TCT fields.

Table 35-6. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	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 19.1.8.2, "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 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 19-1 and Section 5.4.3.7, "Debug Configuration."
	6	DTB	Data buffer and CES transmit cell scratchpad bus placement 0 Reside on the coherent system bus. 1 Reside on the QUICC Engine secondary bus. Note: TCT[DTB] programming must be consistent across all CES transmit channels because they share the same TCELL_TMP_BASE_EXT parameter.
	7	BDB	BD and interrupt queue bus 0 Reside on the coherent system bus. 1 Reside on the QUICC Engine secondary bus. Note: When using AAL5, AAL1 or AAL1 CES in UDC mode, BDs and data should be placed on the same bus (TCT[DTB]=TCT[BDB]).
	8-9	—	Reserved, should be cleared during initialization.
	10-11	ATT	ATM traffic type 00 Peak cell-rate pacing. The host must initialize PCR and the PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and minimum cell rate pacing (UBR+ traffic). The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved.
	12	AVCF	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table, 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the RISC clears the VCON bit. (Bit 13)
	13	VCON	Virtual channel is on Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPS] (stop transmit), the RISC 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 RISC clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 35-6. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-12	—	Reserved, should be cleared during initialization.
	13–15	AAL	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 010 AAL5 —ATM adaptation layer 5. 100 AAL2 —ATM adaptation layer 2. Refer to Chapter 32, "ATM Controller AAL0, AAL1, and AAL5." 101 AAL1_CES —ATM adaptation layer 1 with circuit emulation service 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. Initialize to 0.
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. Initialize to 0.
	8–15	PCR Fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the RISC.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the RISC automatically updates PCR to the ACR value.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC linked channel. Used by the RISC. Initialize to 0 (null pointer).
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	TBD_BASE	TxBD base. Points to the first BD in the channel's TxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zero.
0x1E	0–11		
	12	BNM	Buffer-not-ready interrupt mask. Can be changed on-the-fly. 0 The transmit buffer-not-ready event of this channel is masked. (TBNR event is not sent to the interrupt queue.) 1 The buffer-not-ready event of this channel is enabled.
	13	STPT	Stop transmit. Initialize to 0. When the host sets this bit, the RISC deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. Note that for AAL5 if STPT is set and frame transmission is already started (TCT[INF]=1), an abort indication will be sent (last cell with zero length field).
	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.

35.9.2.1 AAL1 CES Protocol-Specific TCT

Figure 35-24 shows the AAL1 CES protocol-specific transmission connection tables (TCT).

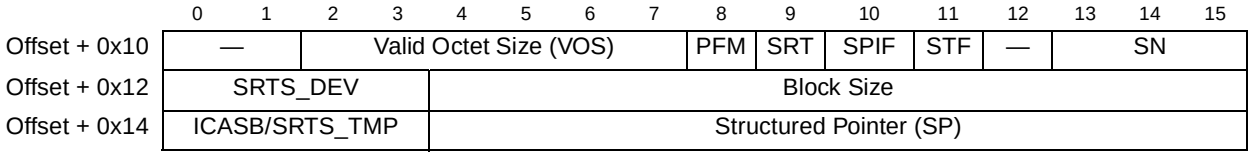


Figure 35-24. AAL1 CES Protocol-Specific TCT

Table 35-7 describes AAL1 CES protocol-specific TCT fields.

Table 35-7. AAL1 CES Protocol-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0-1	—	Reserved, should be cleared during initialization.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1-47 are valid; for structured service, values 1-46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The QUICC Engine block supports SRTS generation using external logic. If this mode is enabled, the QUICC Engine block reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The QUICC Engine block reads the new SRTS from external logic every eight cells. 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPIF	Structured pointer inserted flag. Indicates that a structured pointer has been inserted in the current cycle. The user should initialize this field to zero. **Used by the RISC only**.
	11	STF	Structured format. 0 Unstructured format is used. 1 Structured format is used. Note that although the structured format may be used, if the block size is one, the user should clear STF so that only non P-format AAL1 cells are generated.
	12	—	Reserved, should be cleared during initialization.
	13–15	SN	Sequence number field. Used by the CP to generate and maintain the SN of the cell. Should be cleared initially.
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28:31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum). Note that when working in CAS mode (CESM=1 and CASM=1), the block size should be programmed to the superframe block size plus the size of the CAS block. However, when working in basic mode (CES without CAS: CESM=1 and CASM=0) the block size should be programmed to the number of MCC slots (Nx64) per frame assigned to this channel.

Table 35-7. AAL1 CES Protocol-Specific TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x14	0–3	ICASB/SRTS_TMP	ICASB applies when in CAS mode. Incoming CAS Block. Points to one of the eight available internal CAS block. The starting address of the table is IN_CAS_BLOCK_BASE+ICASB × 32. See Section 35.4.7.1, "CAS Routing Table" for more details. Note that the RCT and TCT use the same CAS routing table (CRT). SRTS_TMP applies when not in CAS mode. Before a cell with SN = 1 is sent, the RISC reads the SRTS code from external SRTS logic, writes it to SRTS_TMP, and then inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the RISC to calculate the structured pointer. Initialize to 0. Structured format only.

35.10 Outgoing CAS Status Register (OCASSR)

Figure 35-25 shows the layout of the outgoing CAS block status register (OCASSR).

	0	7	8	9	10	11	12	13	14	15
Field	—		MCASB7	MCASB6	MCASB5	MCASB4	MCASB3	MCASB2	MCASB1	MCASB0

Figure 35-25. Outgoing CAS Status Register (OCASSR)

This status register contains sticky bits that give the software application an indication that the RISC has modified the associated CAS block. When the ATM receiver operates in core CAS modify mode RCT[CCASM=1], the RISC generates an interrupt and sets the appropriate sticky bit in OCASSR each time an AAL1 cell is received with new signaling information (one or more signaling nibble has changed).

Note that this flag bit stays set until it is cleared by software. If new signaling is received and the relevant sticky bit is already set, the RISC updates the CAS block without generating another interrupt.

Table 35-8 describes OCASSR fields.

Table 35-8. OCASSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared during initialization.
8, 9, 10, 11, 12, 13, 14, 15	MCASB*n*	Modify CAS Block *n*. When the RISC updates this outgoing CAS block, it sets the MCASB*n* sticky bit and generates an interrupt to notify the core that the signaling information has changed in block *n*. Each channel selects the CAS block number in its RCT; see Section 35.9.1.1, "AAL1 CES Protocol-Specific RCT."

35.11 Buffer Descriptors

The AAL1 CES controller operates as a multi-channel protocol, performing simultaneous segmenting and reassembling of transmit and receive data, to and from different sets of memory buffers for all channels. This behavior makes it necessary to have a separate list of BDs for each channel. Each channel is configured with two BD lists: one for transmit and the other for receive operations. The amount of BDs' in the table is defined by the user.

The BD table is a circular list, the last BD in the table holds a wrap indication. When the QUICC Engine block reaches the last BD, it returns to the head of the list. Each BD in the TxBD table points to a buffer to send. At the receive side, the user allocates dedicated buffers to each channel (that is, one BD for each buffer). When the receiver or transmitter completes writing or reading the buffer, it moves to the next buffer in the list and optionally issues an interrupt.

35.11.1 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 RISC reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the RISC 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 RISC continues sending. If the current BD is not ready, the RISC polls the ready bit at the channel rate unless TCT[AVCF] = 1, in which case the RISC removes the channel from the APC and clears TCT[VCON]. The core must issue a new atm transmit command to restart transmission.

Note that when the ATM transmitter is in CES mode, the buffer-not-ready state is ignored by the ATM controller; see Section 35.4.1.2, "TDM-to-ATM."

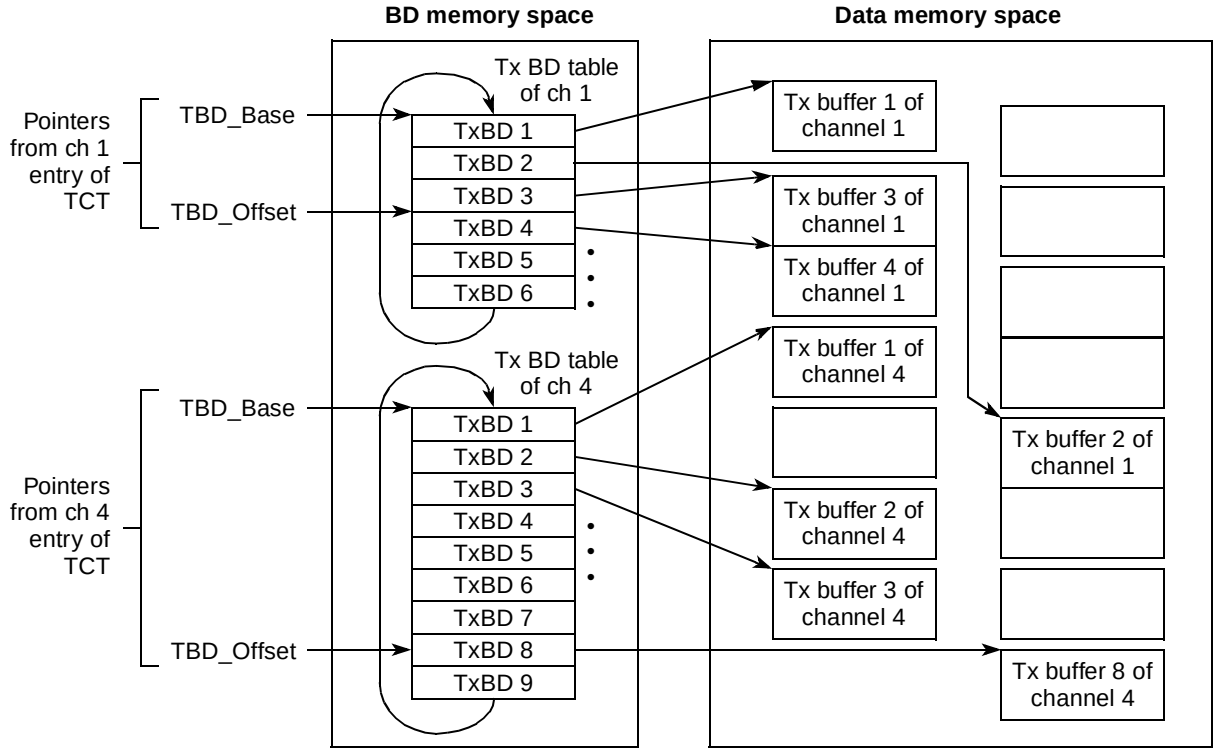


Figure 35-26. Transmit Buffers and BD Table Example

35.11.2 Receive Buffer Operation

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 RISC opens the first BD in the table and

starts filling its associated buffer with received data. When the current buffer is full, the RISC 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 RISC continues receiving. If the BD is not empty, a busy condition has occurred and the ATM receiver optionally issues an interrupt to the event queue.

Note that when the ATM receiver is in CES mode, the buffer-not-ready (busy) state is handled by an automatic slip control mechanism; see Section 35.5, "ATM-to-TDM Adaptive Slip Control."

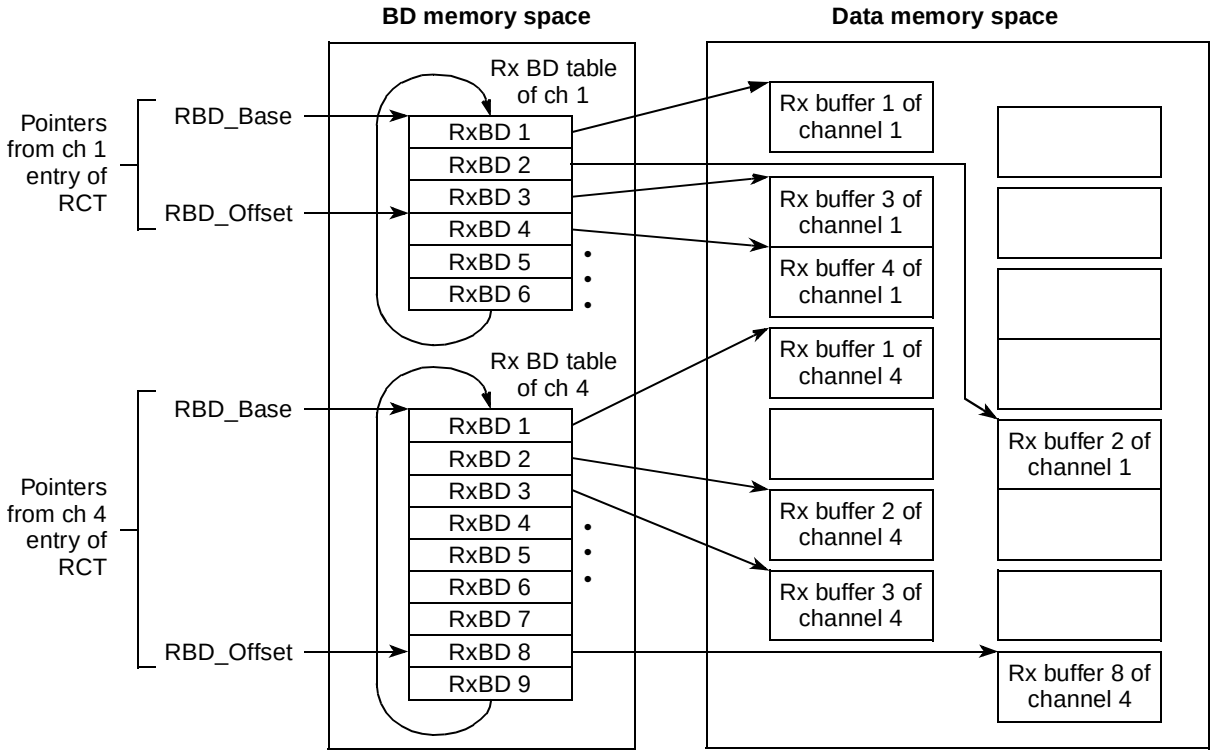


Figure 35-27. Receive Buffers and BD Table Example

35.12 ATM Controller Buffers

Table 35-9 describes properties of the ATM receive and transmit buffers.

Table 35-9. 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
AAL3/4	At least 44 octets (except last buffer in frame)	Double word aligned	At least 44 octets	No requirement
AAL1/ AAL1 CES	Multiple of 8 octets	No requirement	Multiple of 8 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

35.12.1 AAL1 CES RxBD

Figure 35-28 shows the AAL1 CES RxBD.

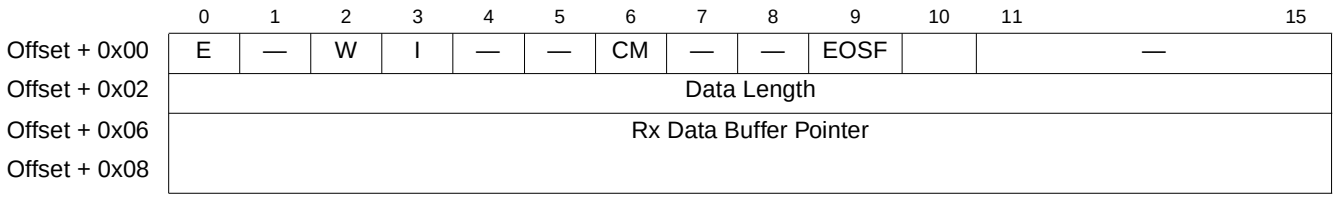


Figure 35-28. AAL1 CES RxBD

Table 35-10 describes AAL1 CES RxBD fields.

Table 35-10. AAL1 CES 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 read or write any fields of this RxBD. The RISC cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the RISC. Once E is set, the core should not write any fields of this RxBD.
	1	—	—
	2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer is used, the RISC receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table overall space is constrained to 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4-5	—	—
	6	CM	Continuous mode 0 Normal operation. 1 The empty bit (RxBD[E]) is not cleared by the RISC after this BD is closed, allowing the associated buffer to be overwritten automatically when the RISC next accesses this BD.
	7-8	—	—
	9	EOSF	End of super frame. CES mode (RCT[CESM=1]) only. 0 No signaling information should be inserted after closing this buffer. 1 When closing this buffer, the ATM receiver unpacks the CAS information from the incoming AAL1 cells and stores it in the internal CAS block. Note that this bit should be set by the user at the end of each super-frame in the common (MCC, ATM) BD table. See Section 35.4.6, “Channel Associated Signaling (CAS) Support.”
	10-15	—	—

Table 35-10. AAL1 CES RxBD Field Descriptions

Offset	Bits	Name	Description
0x02	—	DL	Data length. The number of octets the RISC writes into the buffer once its BD is closed.
0x04	—	RXDBPTR	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

35.12.2 AAL1 CES TxBDs

Figure 35-29 shows the AAL1 CES TxBD.

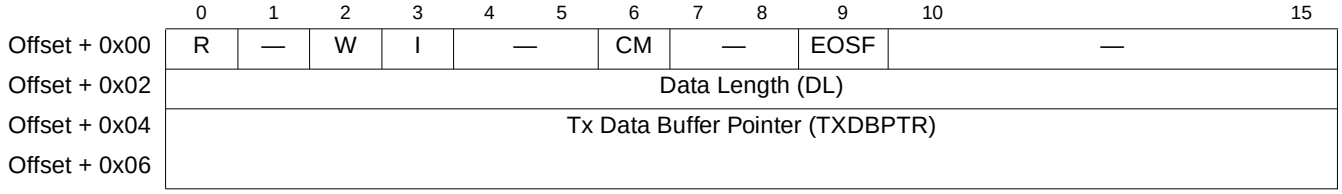


Figure 35-29. AAL1 CES TxBD

Table 35-11 describes AAL1 CES TxBD fields.

Table 35-11. AAL1 CES 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 RISC clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	—
	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 RISC sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	—
	6	CM	Continuous mode 0 Normal operation. 1 The RISC does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the RISC next accesses this BD.
	7–8	—	—

Table 35-11. AAL1 CES TxBD Field Descriptions (continued)

Offset	Bits	Name	Description
	9	EOSF	End of super frame. CES mode (TCT[CESM=1]) only. 0 No signaling information should be inserted after closing this buffer. 1 When closing this buffer the ATM transmitter fetches the CAS information from the internal CAS block and packs it to the outgoing AAL1 cells. Note that this bit should be set by the user at the end of each super-frame in the common (MCC, ATM) BD table. See Section 35.4.6, "Channel Associated Signaling (CAS) Support.". This implies each buffer contains no more then a single super-frame size data.
	10–15	—	—
0x02	—	DL	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the RISC. The value of DL should be greater than zero.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer. The buffer may reside in either internal or external memory. This value is not modified by the RISC.

35.13 AAL1 CES Exceptions

There are four circular interrupt queues available for each channel. The interrupt queue number is programmed in RCT[INTQ] and TCT[INTQ]. Events can be masked by clearing interrupt mask bits in the RCT and TCT.

When one of the AAL1 channels generates an interrupt request, the RISC writes a new entry to the table consisting of the channel's number and a description of the exception. The valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches the entry in which W is set, it returns to the first entry of the queue. Each one-word entry provides detailed interrupt information to the host. More details can be found in Section 29.11, "ATM Exceptions."

35.13.1 AAL1 CES Interrupt Queue Entry

The figure below shows an interrupt queue entry.

	0	1	2	3–7	8	9	10	11	12	13	14	15
Offset + 0x00	**V**	—	**W**	—	**SLIPE**	**SLIPS**	**CASUP**	TBNR	RXF	BSY	TXB	RXB
Offset + 0x02	Channel Code (CC)											

Figure 35-30. AAL1 CES Interrupt Queue Entry

The table below describes interrupt queue entry fields.

Table 35-12. AAL1 CES Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be use by the RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	W	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.

Table 35-12. AAL1 CES Interrupt Queue Entry Field Descriptions (continued)

Offset	Bits	Name	Description
	3–7	—	—
	8	**SLIPE**	**Slip End**.Set when an AAL1 channel exits a slip state (the channel's adaptive counter reaches the ATM_Start threshold and the ATM channel regains its SYNC). At this point the receiver starts to receive the incoming cells. See Section 35.6, "3-Step-SN Algorithm." Note that this interrupt can be masked with RCT[SLIPIM=0]. See Section 35.9.1, "Receive Connection Table (RCT)." This interrupt has an associated channel code.
	9	**SLIPS**	**Slip Start**.Set when an AAL1 channel enters a slip state (the channel's adaptive counter reaches the ATM_Stop threshold or the ATM channel loses its SYNC). At this point the receiver drops incoming cells until the adaptive counter reaches the ATM_Start threshold and the channel is resynchronized. See Section 35.6, "3-Step-SN Algorithm." Note that this interrupt can be masked with RCT[SLIPIM=0]. See Section 35.9.1, "Receive Connection Table (RCT)." This interrupt has an associated channel code.
	10	**CASUP**	**CAS Update Interrupt**. Set when one of the eight outgoing CAS blocks is updated by the RISC. New signaling information has been received within the received AAL1 cells. Note that this interrupt available in CES mode and when RCT[CCASM=1] mode. This interrupt has an associated channel code.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt is issued when the RISC 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.

35.14 AAL1 Sequence Number (SN) Protection Table

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in multi-user RAM and is used for AAL1 only. The table should be initialized according to Figure 35-31.

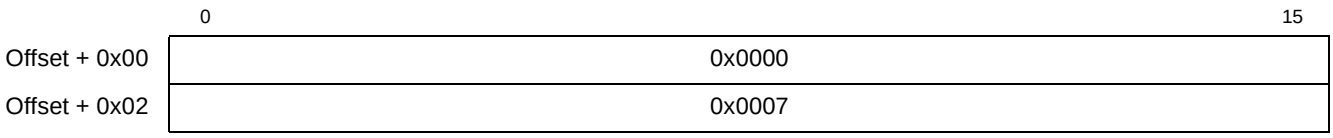


Figure 35-31. AAL1 Sequence Number (SN) Protection Table

Offset + 0x04	0x000D
Offset + 0x06	0x000A
Offset + 0x08	0x000E
Offset + 0x0A	0x0009
Offset + 0x0C	0x0003
Offset + 0x0E	0x0004
Offset + 0x10	0x000B
Offset + 0x12	0x000C
Offset + 0x14	0x0006
Offset + 0x16	0x0001
Offset + 0x18	0x0005
Offset + 0x1A	0x0002
Offset + 0x1C	0x0008
Offset + 0x1E	0x000F

Figure 35-31. AAL1 Sequence Number (SN) Protection Table (continued)

35.15 Internal AAL1 CES Statistics Tables

An AAL1 CES statistics table, shown in Table 35-13, resides in the multi-user RAM and holds AAL1 CES statistics on a per-VC basis. AAL1_Int_STATT_BASE points to the base address of these tables. Each AAL1 channel has its own table with a starting address given by AAL1_Int_STATT_BASE + ATM_CHANNEL# × 8.

Table 35-13. AAL1 CES Multi-user RAM Statistics Table

Offset	Name	Width	Description
0x00	Rx_AAL1_VALID	Hword	16-bit cyclic counter. Counts the total received AAL1 cells delivered to the receive buffers. This counter includes the tag cells (with SCE, SNE).
0x02	Rx_AAL1_BOV	Hword	16-bit cyclic counter. Counts the number of ATM buffer-pre overrun events i.e the ATM write pointer reaches the ATM_STOP threshold.See Section 35.5, “ATM-to-TDM Adaptive Slip Control.”
0x04	Tx_AAL1_VALID	Hword	16-bit cyclic counter. Counts the transmitted AAL1 cells.
0x06	Tx_AAL1_BUN	Hword	16-bit cyclic counter. Counts the number of ATM buffer underrun events. See Section 35.4.1.2, “TDM-to-ATM.”

35.16 External AAL1 CES Statistics Tables

An AAL1 CES statistics table, shown in Table 35-14, resides in the external memory and holds AAL1 CES statistics on a per-VC basis. AAL1_Ext_STATT_BASE points to the base address of these tables. Each AAL1 channel has its own table with a starting address given by AAL1_Ext_STATT_BASE + ATM_CHANNEL# × 16.

Table 35-14. AAL1 CES External Statistics Table

Offset	Name	Width	Description
0x00	Rx_AAL1_LOST	Hword	16-bit cyclic counter. Counts the number of AAL1 lost cells events. See Section 35.6, "3-Step-SN Algorithm."
0x02	Rx_AAL1_MISS	Hword	16-bit cyclic counter. Counts the number of AAL1 misinserted events. See Section 35.6, "3-Step-SN Algorithm."
0x04	Rx_AAL1_SCE	Hword	16-bit cyclic counter. Counts the number of AAL1 sequence errors i.e the expected SC is not match with the received one (ESC!= RSC). See Section 35.6, "3-Step-SN Algorithm."
0x06	Rx_SNP_Error	Hword	16-bit cyclic counter. Counts the number of ATM cell that received with SNP error. (AAL1 PDU Header Error)
0x08	Rx_AAL1_SPE	Hword	16-bit cyclic counter. Counts the number of structured pointer error events (i.e parity error, Tag cell or pointer mismatch). See Section 35.7, "Pointer Verification Mechanism."
0x0A	Rx_ReSYNC	Hword	16-bit cyclic counter. Counts the number of AAL1 resynchronized events: pointer reframes, slip events, two consecutive cells with errors (SNE, SCE, Tag…), and two consecutive pointers with errors (parity error or pointer mismatch).
0x0C–0x0E	—	Word	Reserved, should be cleared during initialization.

Note that both the internal and the external statistics tables should be cleared by the user.

35.17 CES-Specific Additions to the MCC

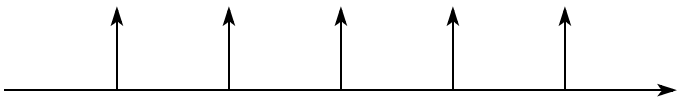
Additions to the MCC global parameter RAM and modifications to the channel-specific CHAMR and INTMASK registers have been implemented to support CES operation. Refer to Section 34.2.2.3, "MCC Parameters for AAL1 CES Usage," Section 34.2.2.3.3, "Interrupt Circular Table Entry and Interrupt Mask (INTMSK)—AAL1 CES," and Table 34-8 for more information.

35.18 Application Considerations

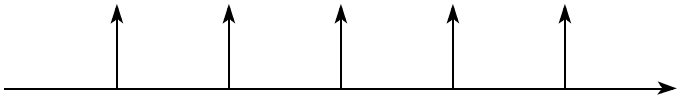
- The buffer size (MCC [MRBLR]) must be a multiple of 8 octets.
- The CAS buffer operates in continuous mode with newer signaling information continuously overwriting older signaling.
- The CES application does not use super-frame synchronization for the data flow in ATM-to-TDM and TDM-to-ATM interworking. However, for the signaling flow, the QUICC Engine block uses the super-frame sync signal to know when to supply the signaling information to the external framer. The external framer then places the signaling information at the appropriate position in the super frame. See Section 35.4.6, "Channel Associated Signaling (CAS) Support."
- Simple external logic is needed to synchronize the MCC-to-framer serial connection. When going from TDM-to-ATM, the CAS information should be read by the MCC on the 24th frame of a super frame.
- The adaptive rate FIFO method can be implemented by the core by calculating the difference between the ATM and MCC pointers.

- When going from TDM-to-ATM, the ATM channel should be programmed to work at a higher rate than the MCC super-channel rate. We expect that the jitter caused by the APC traffic reshaping will depend on the ATM channel rate (PCR, PCR_FRACTION). Figure 35-32 illustrates this timing issue. See also Section 35.4.1.2, "TDM-to-ATM."

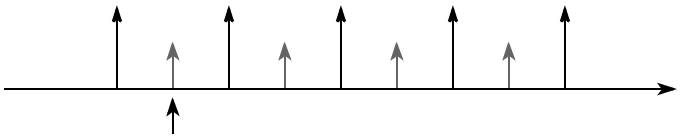
MCC timing:
BDs are ready to be transmitted at the rate shown below.



ATM timing:
The optimal ATM channel rate would match the MCC super channel rate exactly.



ATM timing (adjusted to higher rate):
If PCR and PCR_FACTION cannot provide the exact MCC super channel rate, the ATM channel should be programmed to a higher rate to avoid the MCC buffer-not-ready state. It is recommended that the ATM rate be double that of the MCC.



Extra request at the higher rate to compensate for jitter.

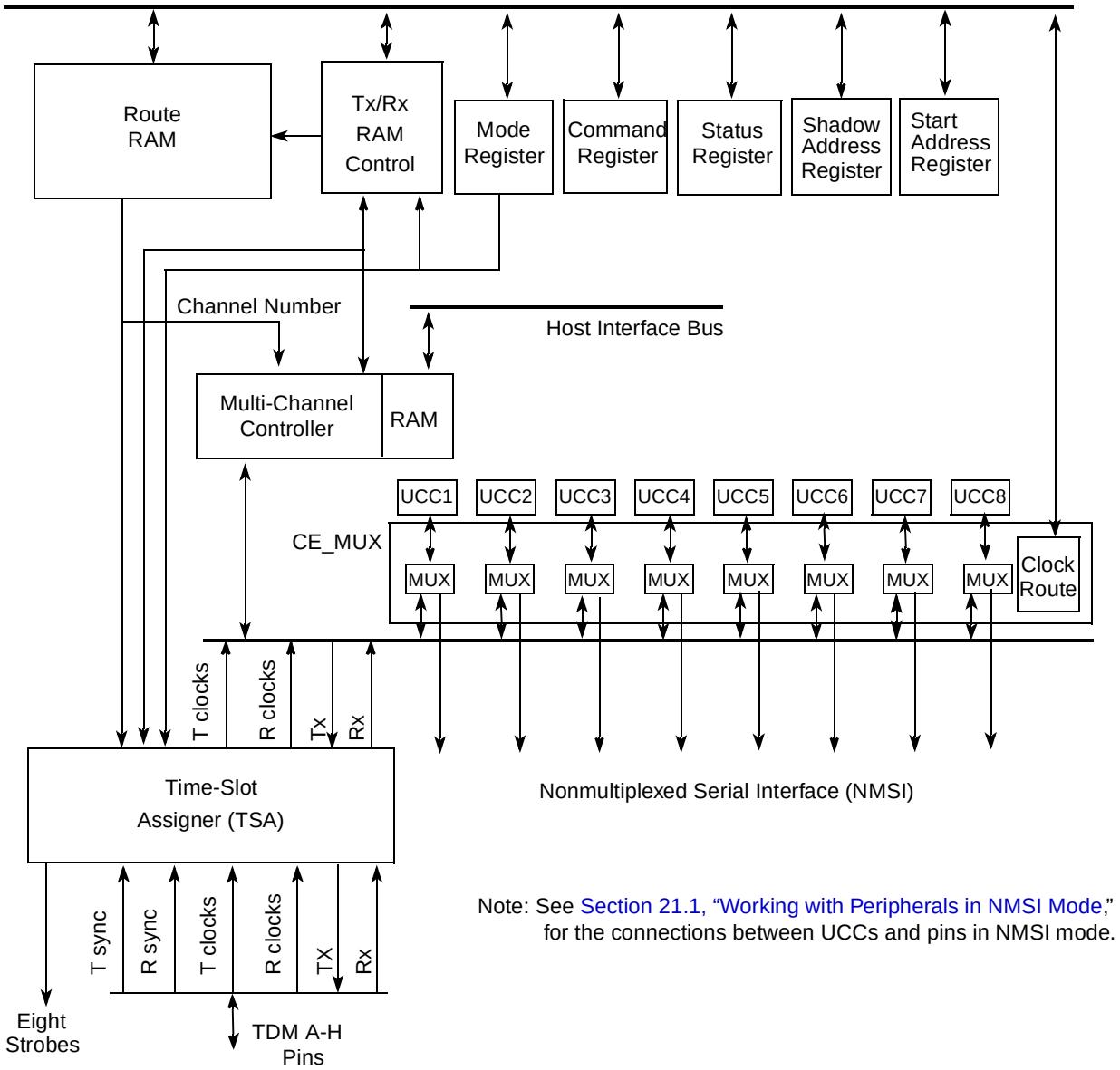
Note that some of the extra requests will not be needed (buffer-not-ready) and will be ignored by the ATM controller.

Figure 35-32. TDM-to-ATM Timing Issue

Chapter 36
Serial Interface with Time-Slot Assigner

36.1 Serial Interface Block Diagram

The serial interface (SI) block diagram is shown in Figure 36-1.



Note: See Section 21.1, "Working with Peripherals in NMSI Mode," for the connections between UCCs and pins in NMSI mode.

Figure 36-1. SI Block Diagram

NOTE

The QUICC Engine MUX and the multi-channel controller are not part of the SI but they are shown because SI is tightly coupled with them.

36.2 Overview

The serial interface (SI) manages the routing of eight time-division multiplexing (TDM) lines to the QUICC Engine block serial drivers, the MCC and 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 eight time-slot assigner (TSA) modules, each handling one TDM line and is connected to a dedicated MCC router. Each TSA can also 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 36-2 through Figure 36-5. TSA programming is independent of the protocol used by the MCC or 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.

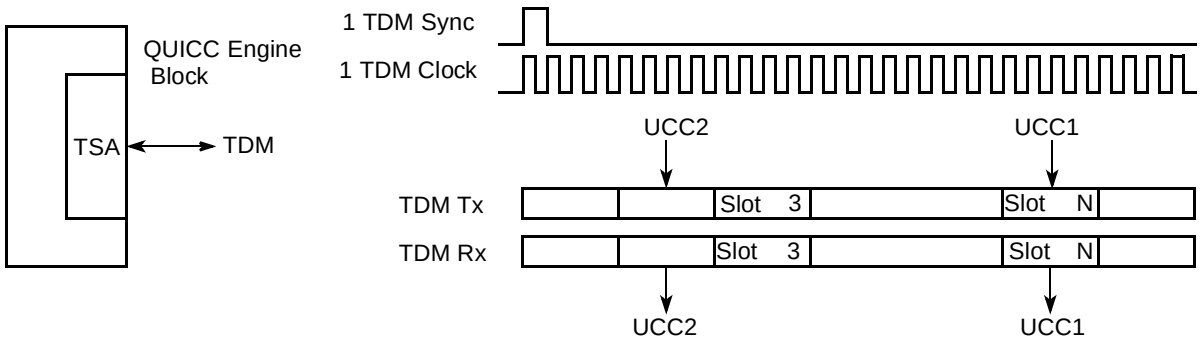


Figure 36-2. Simple TDM Example

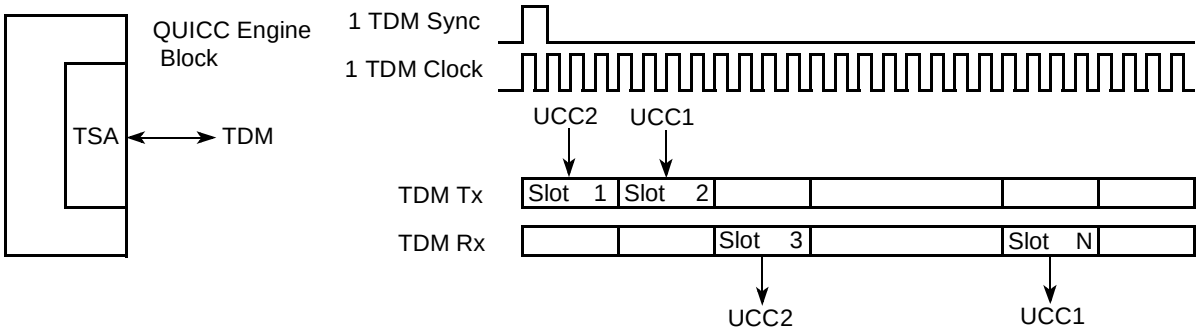


Figure 36-3. TDM Example—Different Tx and Rx Routing

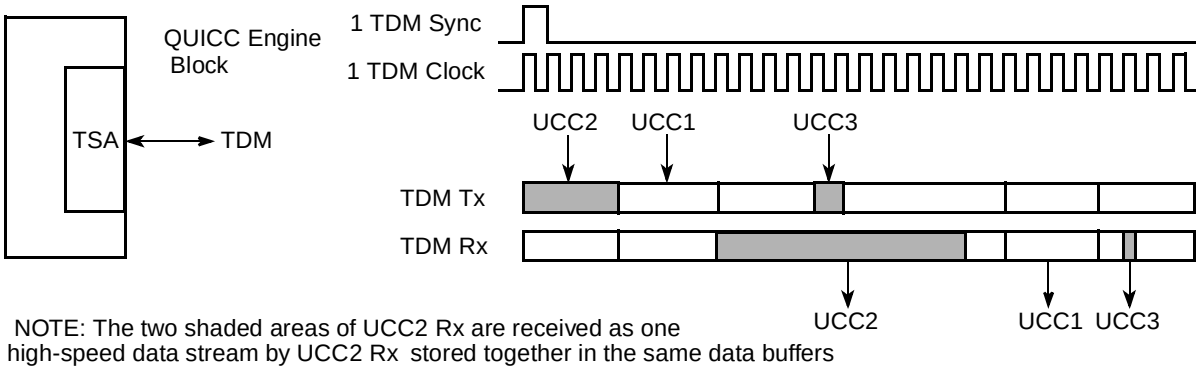


Figure 36-4. TDM Example—Multiple Time Slot Per Channel With Varying Sizes of Time Slot

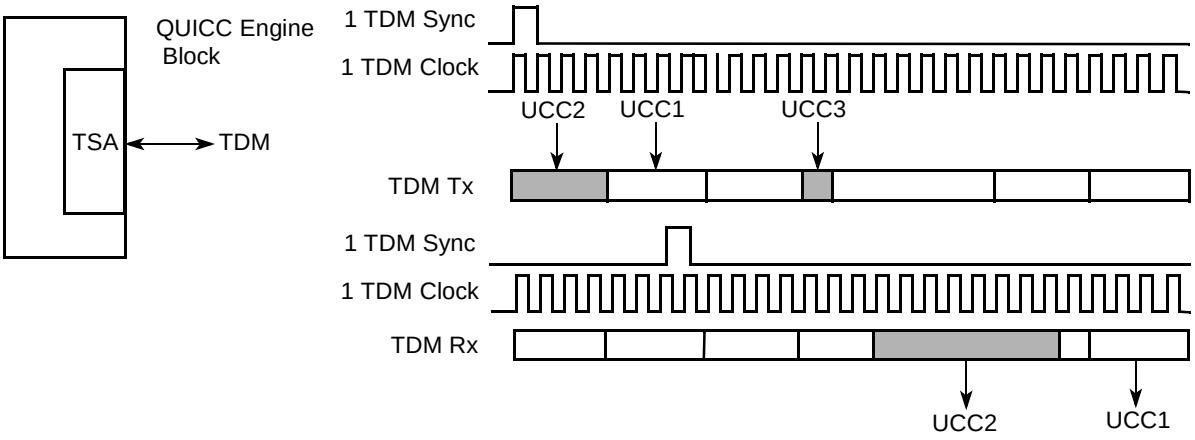
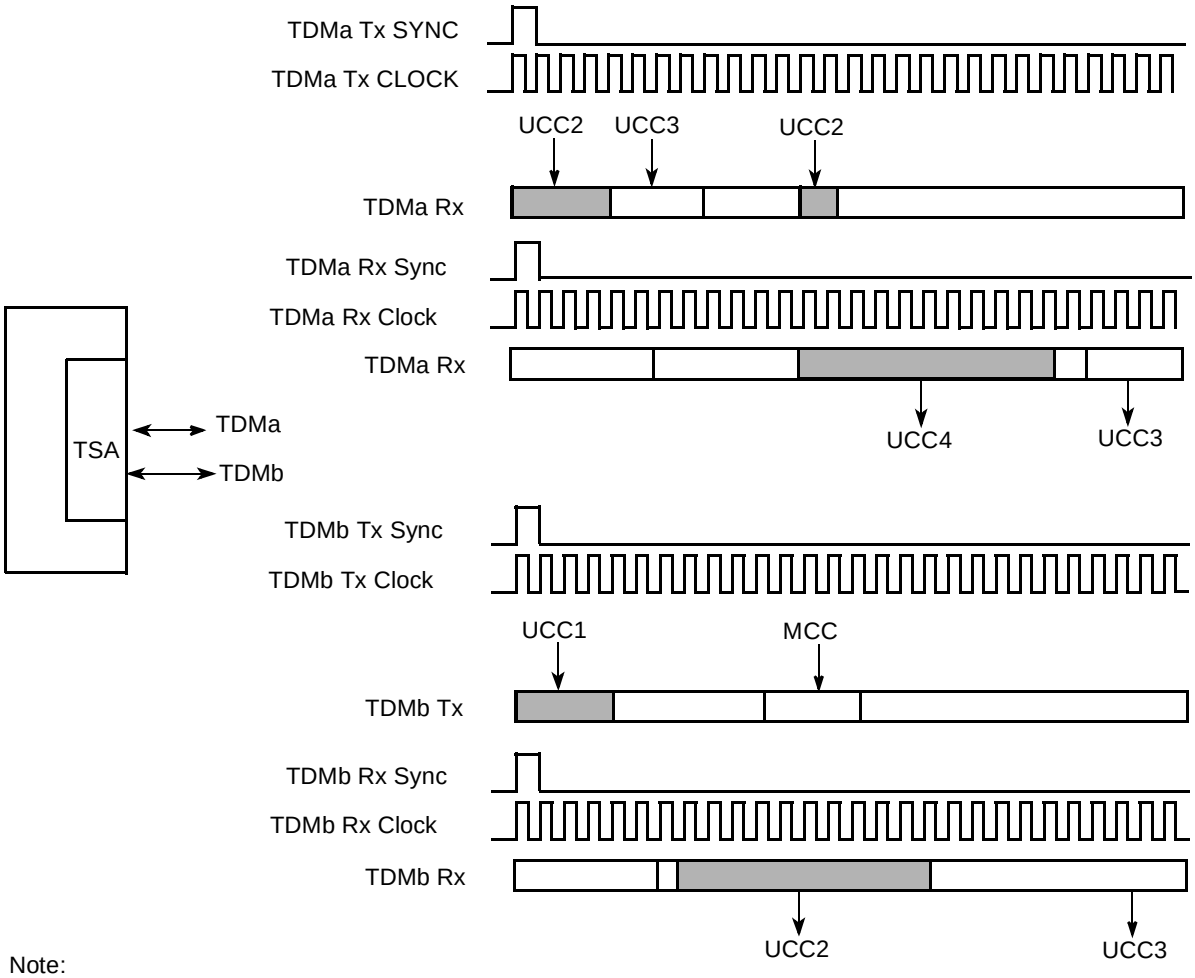


Figure 36-5. TDM Example—Totally Independent Rx and Tx

At its most flexible, the TSA can provide eight separate TDM channels, each with independent receive and transmit routing assignments and independent sync pulse and clock inputs. Thus, the TSA can support sixteen, independent, half-duplex TDM sources, eight in reception and eight in transmission, using eight sync inputs and eight clocks each. A dual-channel example is shown in Figure 36-6.



Note:
UCCs can receive on one TDM and transmit on another (UCC2 and UCC3).

Figure 36-6. Dual TDM Channel Example

In addition to channel programming, the TSA supports up to eight 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 and MCC. 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 MCC or 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.

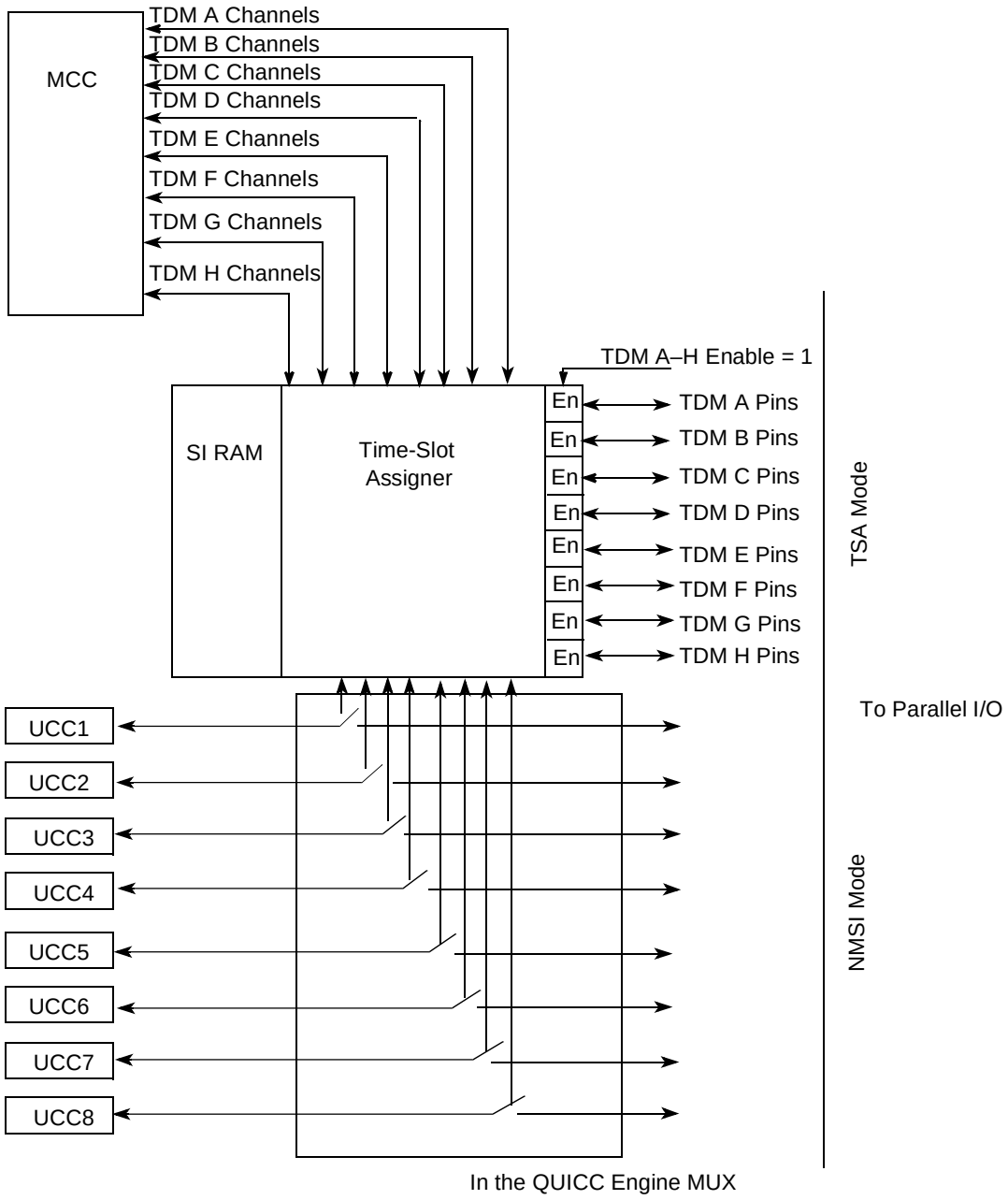
The SI supports two testing modes—echo and loopback.

- 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 eight TDM channels and to all MCC and UCC's interfaces independently.

36.2.1 Enabling Connections to TSA

Each UCC can be independently enabled to connect to TSA or dedicated external pins (NMSI). The MCC can be connected to any of the eight TDMs with different numbers of channels. The eight TDMs are connected to eight independent TDM interfaces. The connection between the TSA and the serial interfaces is shown in Figure 36-7. 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 UCC's and pins in NMSI mode are not shown in this figure.

Figure 36-7. Enabling Connections to the TSA

36.3 Features

The SI includes the following features:

- Can connect to eight 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.
- Eight programmable strobe outputs and eight 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 0–255 time slots to MCC or to any of 8 UCCs.

For the MCC route, the SI performs the following features:

- Up to 256 independent communication channels (on 256 time slots).
- Arbitrary mapping of any of the 256 channels to any TDM time slots.
- Can connect up to eight independent TDM channels. Each TDM channel can support up to 256 channels (all 8 channels can support up to 256 channels together).
- Independent mapping for receive/transmit.
- Individual channel echo or loop mode.
- Switch RxD and TxD lines per bit.
- Global echo or loopback mode through the SI.
- Periodic Multiframe support.[1] An efficient way of interleaving two independent frames and reduce the usage of the shadow RAM.

NOTE: On Backward-Compatibility

Backward-compatibility for the programming model is kept in the SI except for the following cases:

1. Feature available for TDM E only.

- TDMs E-H different memory map.
- 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 MCC and UCC.
- Multiframe for all TDMs.
- 256 channels for MCC
- Eight programmable strobe outputs (was only four strobe outputs)
- High-speed operation for UCC entries (the minimal clock ratio between serial and CE clock is 1:8, the maximal clock ratio is 1:16). See Section 36.6.14, "SI Speed Register (SISPD)."

36.4 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.

36.5 SI External Signal Descriptions

This section defines the SI to TDM I/O pins. SI external signal list is shown in Table 36-1.

Table 36-1. SI External Signal Table

Name	Function	I/O
SI:STRB[1:8]	SI strobe signal	O
TDM[A:H]:CLKO	clock out for TDM[A:H]	O

Table 36-1. SI External Signal Table

Name	Function	I/O
TDM[A:H]:RSYNC	Receive data sync TDM[A:H]	I
TDM[A:H]:RXD[0:3]	Receive data TDM[A:H]	I
TDM[A:H]:TSYNC	Transmit data sync TDM[A:H] or IDL grant.	I
TDM[A:H]:TXD[0:3]	Transmit data TDM[A:H]	O
TDM[A:H]:RQ	IDL request permission to transmit on D channel	O

36.5.1 SI Detailed Signal Descriptions

Detailed information about SI external signal list is shown in Table 36-2.

Table 36-2. Interface A—Detailed Signal Descriptions

Signal	I/O	Description	
SI:STRB[1:8]	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:H]:CLKO	O	clock out for TDM[A:H].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:H]:RSYNC	I	Receive frame sync from TDM [A:H].	
		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.
TDM[A:H]:RxD[0:3]	I	Receive data from TDM[A:H]**.**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.

Table 36-2. Interface A—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDM[A:H]:TSYNC	I	Transmit frame sync to TDM [A:H]. 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:H]:TxD[0:3]	O	transmit data to TDM [A:H].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:H]: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.

36.6 SI Register Definition

The list of all internal SI registers is shown in Table 36-3.

Table 36-3. 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	36.6.4/36-18
0x02	SI TDM B mode register (SIBMR)	16 bits	R/W	36.6.4/36-18
0x04	SI TDM C mode register (SICMR)	16 bits	R/W	36.6.4/36-18
0x06	SI TDM D mode register (SIDMR)	16 bits	R/W	36.6.4/36-18
0x08	SI global mode register high (SIGLMRH)	8 bits	R/W	36.6.2/36-16

Table 36-3. SI Register Summary (continued)

Offset from SI1_base	Registers Name	Size	Access	Section/Page
0x0A	SI command register high (SICMDRH)	8 bits	R/W	36.6.7/36-26
0x0C	SI status register high (SISTRH)	8 bits	R	36.6.9/36-28
0x0E	SI RAM shadow address register high (SIRSRH)	16 bits	R/W	36.6.9/36-28
0x10	SI RAM counter Tx TDM A (SITARC)	8 bits	R	36.6.12/36-32
0x11	SI RAM counter Tx TDM B (SITBRC)	8 bits	R	36.6.12/36-32
0x12	SI RAM counter Tx TDM C (SITCRC)	8 bits	R	36.6.12/36-32
0x13	SI RAM counter Tx TDM D (SITDRC)	8 bits	R	36.6.12/36-32
0x14	SI RAM counter Rx TDM A (SIRARC)	8 bits	R	36.6.12/36-32
0x15	SI RAM counter Rx TDM B (SIRBRC)	8 bits	R	36.6.12/36-32
0x16	SI RAM counter Rx TDM C (SIRCRC)	8 bits	R	36.6.12/36-32
0x17	SI RAM counter Rx TDM D (SIRDRC)	8 bits	R	36.6.12/36-32
0x20	SI TDM E mode register (SIEMR)	16 bits	R/W	36.6.4/36-18
0x22	SI TDM F mode register (SIFMR)	16 bits	R/W	36.6.4/36-18
0x24	SI TDM G mode register (SIGMR)	16 bits	R/W	36.6.4/36-18
0x26	SI TDM H mode register (SIHMR)	16 bits	R/W	36.6.4/36-18
0x28	SI global mode register low (SIGLMRL)	8 bits	R/W	36.6.3/36-17
0x2A	SI command register low (SICMDRL)	8 bits	R/W	36.6.8/36-27
0x2C	SI status register low (SISTRL)	8 bits	R	36.6.10/36-29
0x2E	SI RAM shadow address register low (SIRSRL)	16 bits	R/W	36.6.6/36-25
0x30	SI RAM counter Tx TDM E (SITERC)	8 bits	R	36.6.12/36-32
0x31	SI RAM counter Tx TDM F (SITFRC)	8 bits	R	36.6.12/36-32
0x32	SI RAM counter Tx TDM G (SITGRC)	8 bits	R	36.6.12/36-32
0x33	SI RAM counter Tx TDM H (SITHRC)	8 bits	R	36.6.12/36-32
0x34	SI RAM counter Rx TDM E (SIRERC)	8 bits	R	36.6.12/36-32
0x35	SI RAM counter Rx TDM F (SIRFRC)	8 bits	R	36.6.12/36-32
0x36	SI RAM counter Rx TDM G (SIRGRC)	8 bits	R	36.6.12/36-32
0x37	SI RAM counter Rx TDM H (SIRHRC)	8 bits	R	36.6.12/36-32
0x40	SI multiframe limits register for TDM E (SIMLE)	32 bits	R/W	36.6.1/36-12
0x44	SI enhanced SDM (SIENS)	8 bits	R/W	36.6.13/36-32
0x46	SI speed mode register (SISPD)	8 bits	R/W	36.6.14/36-32
0x47	SI TX Clock Edge Invert register (SITXCEI)	8 bits	R/W	36.6.15/36-33
0x48	SI multiframe limits register for TDMA (SIMLA)	32 bits	R/W	36.6.11/36-29
0x4C	SI multiframe limits register for TDM B (SIMLB)	32 bits	R/W	36.6.11/36-29
0x50	SI multiframe limits register for TDM C (SIMLC)	32 bits	R/W	36.6.11/36-29

Table 36-3. SI Register Summary (continued)

Offset from SI1_base	Registers Name	Size	Access	Section/Page
0x54	SI multiframe limits register for TDM D (SIMLD)	32 bits	R/W	36.6.11/36-29
0x58	SI multiframe limits register for TDM E (SIMLE) [1]	32 bits	R/W	36.6.11/36-29
0x5C	SI multiframe limits register for TDM F (SIMLF)	32 bits	R/W	36.6.11/36-29
0x60	SI multiframe limits register for TDM G (SIMLG)	32 bits	R/W	36.6.11/36-29
0x64	SI multiframe limits register for TDM H (SIMLH)	32 bits	R/W	36.6.11/36-29

[1] SIMLE can be accessed by offset 0x40 or by offset 0x58.

36.6.1 SI RAM Entry

SI RAM is shown in Figure 36-8.

Access: Read/Write

	0	1	2	3	4	5	6	7	10	11	13	14	15
R W	MCC	SWTR	SSEL 1	SSEL 2	SSEL 3	SSEL 4	SGS	CSEL		CNT		BYT	LST
Reset	All zeros												

Figure 36-8. SI RAM Entry for UCC

Table 36-4 describes the SI RAM fields.

Table 36-4. SI RAM Entry Field Description, MCC = 0

Bits	Name	Description
0	MCC	The entry controls the functionality of the other bits in the SI RAM entry. 0 The entry refers to UCC. 1 The entry refers to the MCC.
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.

Table 36-4. SI RAM Entry Field Description, MCC = 0 (continued)

Bits	Name	Description
2–5	SSELx	Strobe select. There are eight 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 eight strobe pins for all sixteen 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. **Note:** Strobes combination can be only of the same SGS group, e.g. there can not be a combination of strobe1 and strobe6. If SGS is 0: 1xxx Strobe 1. x1xx Strobe 2. xx1x Strobe 3. xxx1 Strobe 4. If SGS is 1: 1xxx Strobe 5. x1xx Strobe 6. xx1x Strobe 7. xxx1 Strobe 7.
6	SGS	Strobe group select. Select between four first strobes and four last strobes. 0 Four first strobes are selected. 1 Four last strobes are selected
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 The bit/byte group is routed to ucc6. 0011 The bit/byte group is routed to ucc7. 0100 The bit/byte group is routed to ucc8. 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, i.e. CNT=1 equals 4 bits in parallel.

Table 36-4. SI RAM Entry Field Description, MCC = 0 (continued)

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.

NOTE: On Backward-Compatibility

As in PowerQUICC II Pro, there is a minimal restriction of two entries per frame. There is also a minimal restriction of two bits per UCC entry if CDP or CTSP modes are set.

SWTR example is shown in Figure 36-9. 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.

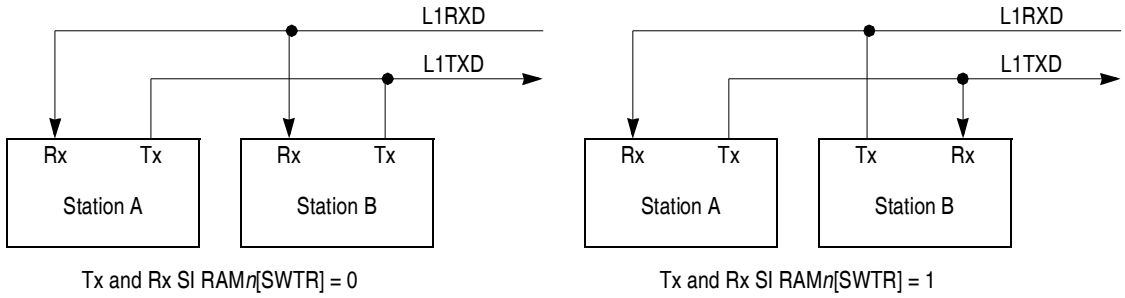


Figure 36-9. Using the SWTR Feature

SI RAM entry register with MCC = 1, is shown in Figure 36-10.

Access: Read/Write

	0	1	2	3	10	11	13	14	15
R W	MCC	LOOP/ ECHO	SUPER	MCSEL		CNT		BYT	LST
Reset	All zeros								

Figure 36-10. SI RAM Entry for MCC

Table 36-5 describes SI RAM entry with MCC = 1 fields.

Table 36-5. SI RAM Entry with MCC = 1

Bits	Name	Description
0	MCC	The entry controls the functionality of the other bits in the SI RAM entry. 0 The entry refers to UCC. 1 The entry refers to the MCC.
1	LOOP/ECHO	Channel loopback or echo. In the receive SI RAM, this bit selects loopback mode for this MCC channel. The channel's transmit data is sent to both the receiver's input and to the data output line. In the transmit SI RAM, this bit selects echo mode for this MCC channel. The channel's receive data is sent both to the transmitter's line and to the receiver input. **Note:** For loopback operation, set LOOP/ECHO=1 in the Rx and LOOP/ECHO=0 in the Tx. For echo operation, set LOOP/ECHO=0 in the Rx and LOOP/ECHO=1 in the Tx. **Note:** To use the loop/echo modes program the receive and transmit SIx RAMs identically, except that the LOOP/ECHO field. **Note:** To switch between Rx and Tx pin functionality set LOOP/ECHO=1 in both the Rx and Tx entries, see Section 36.6.1, "SI RAM Entry," SWTR. 0 Normal mode of operation. 1 Operation depends on the above configurations:
2	SUPER	MCC super channel enable 0 The current entry refers to a super channel.
3–10	MCSEL	MCC channel select. Indicates the MCC channel the bit/byte group is routed to 0000_0000 selects channel 0; 1111_1111, selects channel 255.
11–13	CNT	If SUPER = 0 (normal mode), CNT indicates the number of bits/bytes (according to the BYT bit) that the routing select of this entry controls. 000 = 1 bit/byte; 111 = 8 bits/bytes. If SUPER = 1 (MCC super channel), CNT and BYT together indicate whether the current entry is the first byte of the MCC super channel. BYT = 1—The current entry is the first byte of this MCC super channel. BYT = 0—The current entry is not the first byte of this MCC super channel. CNT: The length may be one of the following: 5,6,7,8 or 16 bits. Refer to Table 36-6 for more information. **Note:** when SUPER = 1, all SI RAM entries relating to MCC must be of the same length.
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.

36.6.1.1 MCC Superchannel Coding

Table 36-6 shows coding of MCC command entry for different cases.

Table 36-6. Super Channel Coding

SuperChannel Bit Entry bit[2]	CNT Entry Bits[11:13]	Bit/Byte Entry Bit[14]	Description
1	000	1	First byte in super channel
1	111	0	Not first byte in super channel
1	001	1	First 2 bytes in super channel
1	001	0	Not first 2 bytes in super channel (SuperChannel does not use a single bit)
1	100	1	First 5 bits in super channel
1	100	0	Not First 5 bits in super channel (SuperChannel does not use 5 bytes)
1	101	1	First 6 bits in Super channel
1	101	0	Not first 6 bits in super channel (SuperChannel does not use 6 bytes)
1	110	1	First 7 bits in super channel
1	110	0	Not First 7 bits in super channel (SuperChannel does not use 7 bytes)

Superchannel is not a protocol dependant mode, rather a performance enhancing mode.

Superchannelling is used for sending multiple back-to-back timeslots from the same MCC channel. A single MCC channel is the combination of one MCC channel and one set of related channel parameters and buffer descriptors. A superchannel is the combination of multiple MCC TX channels and one set of MCC channel's parameters and buffer descriptors to manage this group of channels. This provides the ability to construct a larger overall channel than that of a single normal MCC TX channel. Superchannel makes use of HW resources of unused channels to enhance its performance, the user may not use the channels in the superchannel for any other purpose nor use the same channel twice in the construction of a superchannel.

The length of superchannelled timeslot may be one of the following sizes 5, 6, 7, 8 or 16 bits.

36.6.2 SI Global Mode Register High (SIGLMRH)

SI Global Mode Register High is shown in Figure 36-11.

Offset 0x08 Access: Read/Write

	0	1	2	3	4	5	6	7
R	STZD	STZC	STZB	STZA	END	ENC	ENB	ENA
W								
Reset	All zeros							

Figure 36-11. SI Global Mode Register High (SIGLMRH)

Table 36-7 describes SI Global Mode Register fields.

Table 36-7. 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.

36.6.3 SI Global Mode Register Low (SIGLMRL)

SI Global Mode Register Low is shown in Figure 36-12.

Offset 0x28 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	STZH	STZG	STZF	STZE	ENH	ENG	ENF	ENE
Reset	All zeros							

Figure 36-12. SI Global Mode Register Low (SIGLMRL)

SI Global Mode Register low is register for the low four TDMs (TDM E-TDM H). It is shown in Table 36-7.

Table 36-8. SI Global Mode Register Low (SIGLMRL)

Bits	Name	Description
0–3	STZx	Program L1TXD to zero for TDM e, f, g or h 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 TDM*x.* **Note:** 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.

36.6.4 SI Mode Register (SI*x*MR)

There are eight SI*x*MR registers, one for each TDM. The register is shown in Figure 36-13.

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 36-13. SI Mode Register (SI*x*MR)

NOTE

Adjacent sync signals should have minimum delay between them of at least one clock cycle.

Table 36-9 describes SI Mode Register fields.

Table 36-9. 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 i.e. 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

Table 36-9. SI Mode Register Description (continued)

Bits	Name	Description
4–5	SDM*x*	SI Diagnostic Mode for all eight 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 eight 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.
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.

Table 36-9. SI Mode Register Description (continued)

Bits	Name	Description
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 36-14.

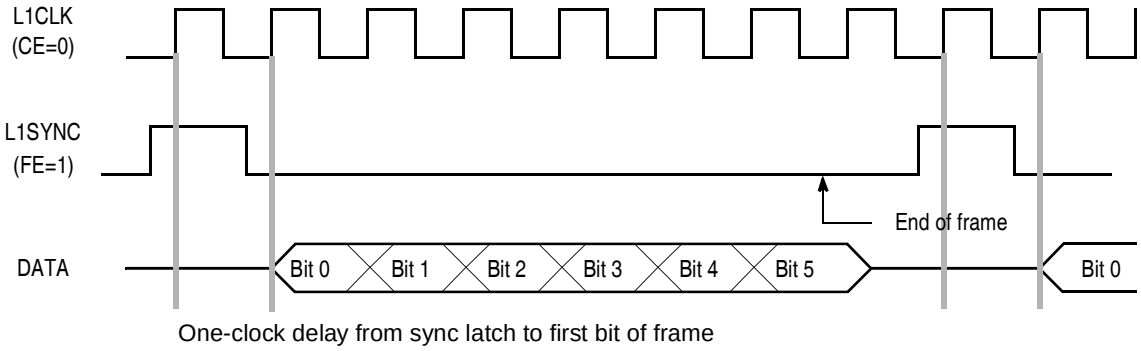


Figure 36-14. One-Clock Delay from Sync to Data (SI*x*MR[*n*FSD] = 01)

The elimination of the single-clock delay shown in Figure 36-15. Because SI*x*MR[*n*FSD] = 00 there is no delay between L1SYNC and the first bit of the frame.

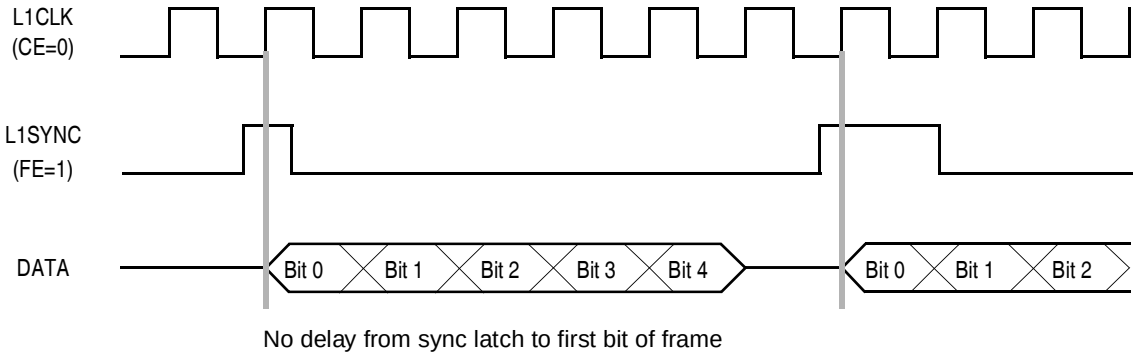


Figure 36-15. No Delay from Sync to Data (SI*x*MR[*n*FSD] = 00)

The difference between SI*x*MR[FE] = 0 and SI*x*MR[FE] = 1 is shown in Figure 36-16. In this example, SI*x*MR[CE] = 1 and there is 1-bit frame delay (SI*x*MR[*n*FSD] = 01).

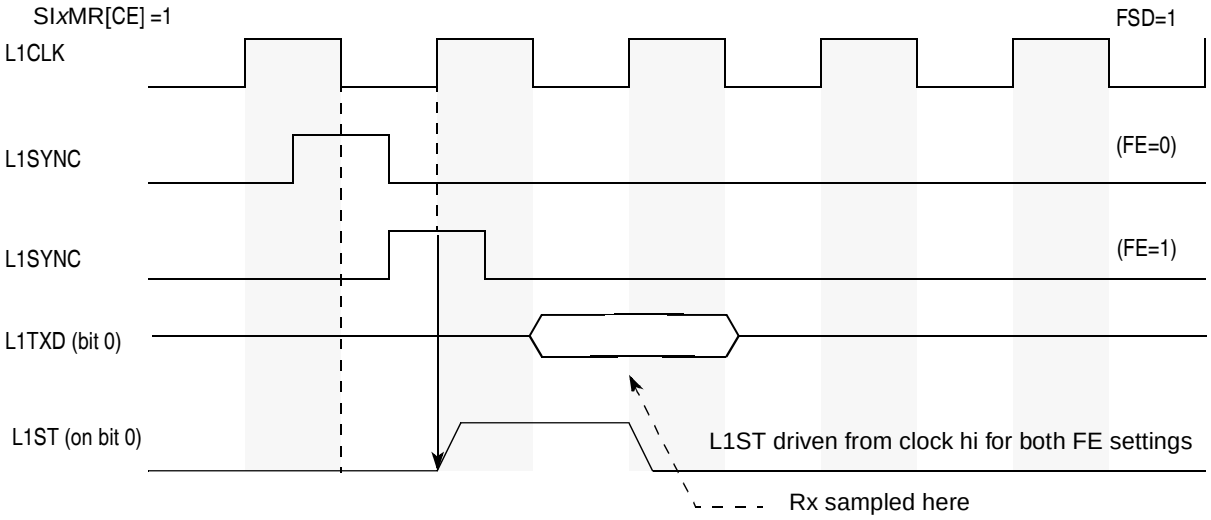


Figure 36-16. Falling Edge Effect when SI*x*MR[CE] = 1 and SI*x*MR[*n*FSD] = 01

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 36-17.

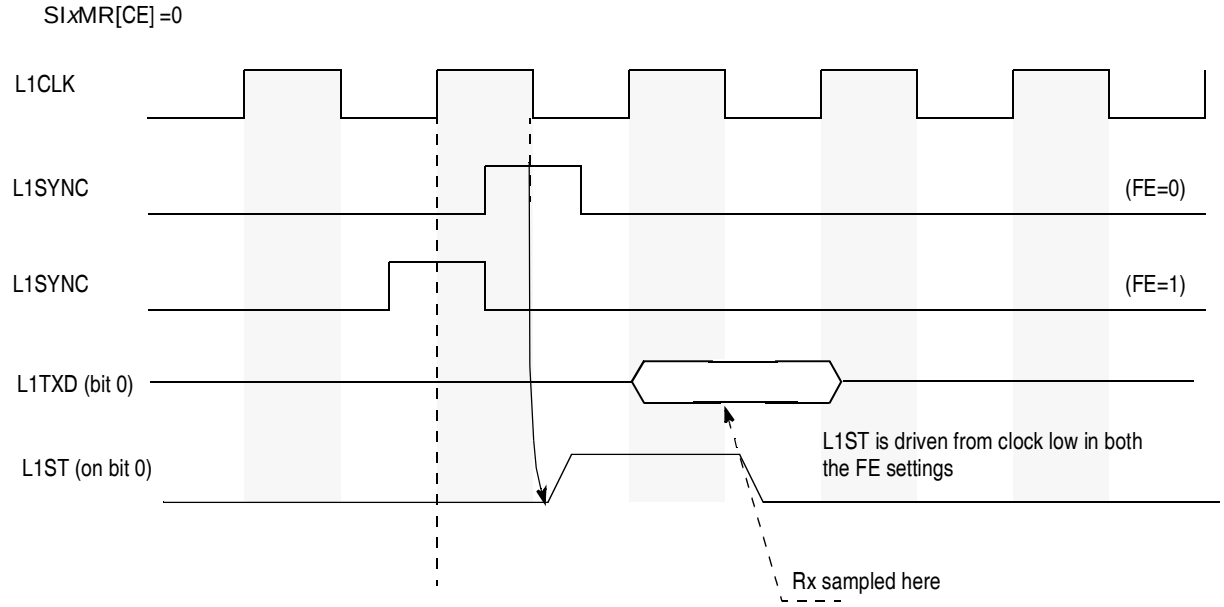


Figure 36-17. Falling Edge Effect when SI*x*MR[CE] = 0 and SI*x*MR[*n*FSD] = 01

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.

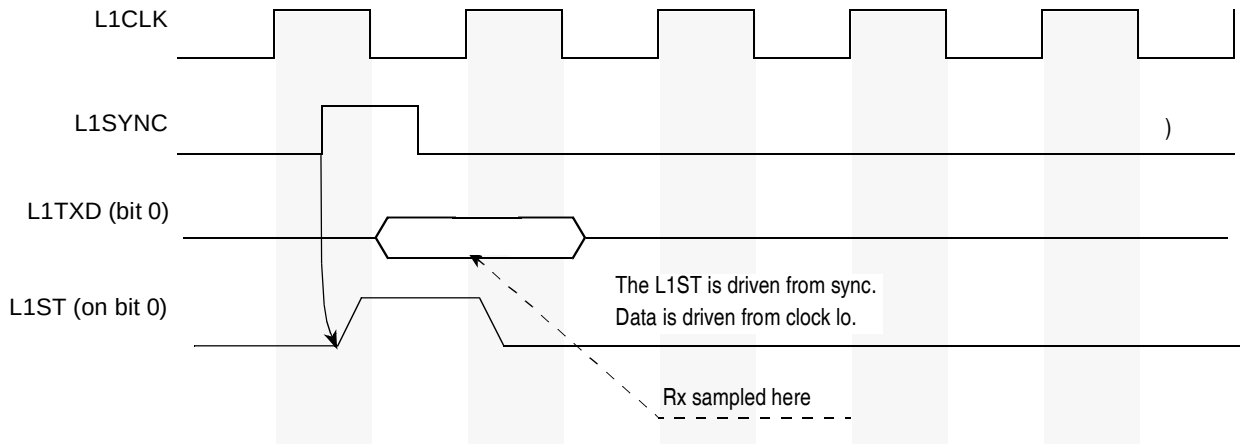


Figure 36-18. Falling Edge Effect When SI*x*MR[CE] = 1, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

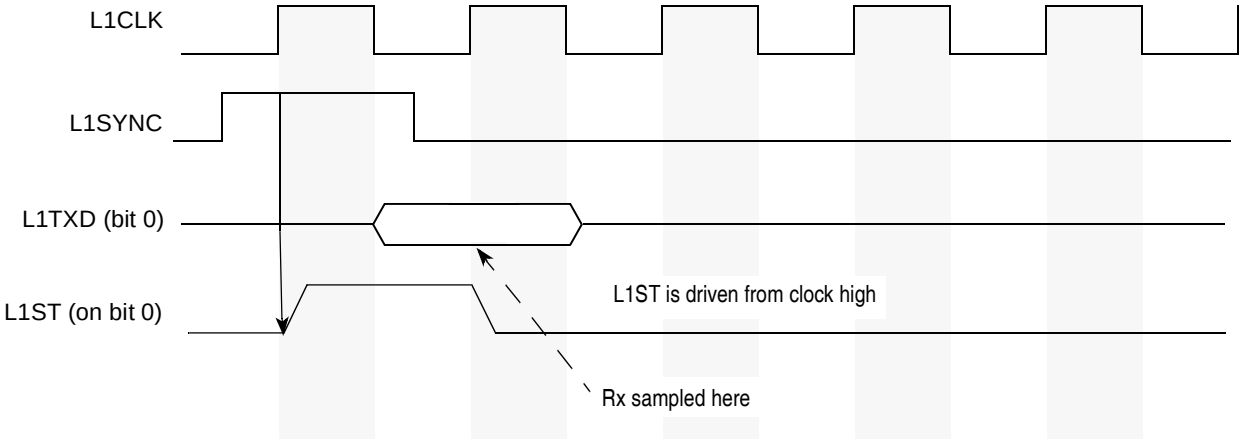


Figure 36-19. Falling Edge Effect When SI*x*MR[CE] = 1, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

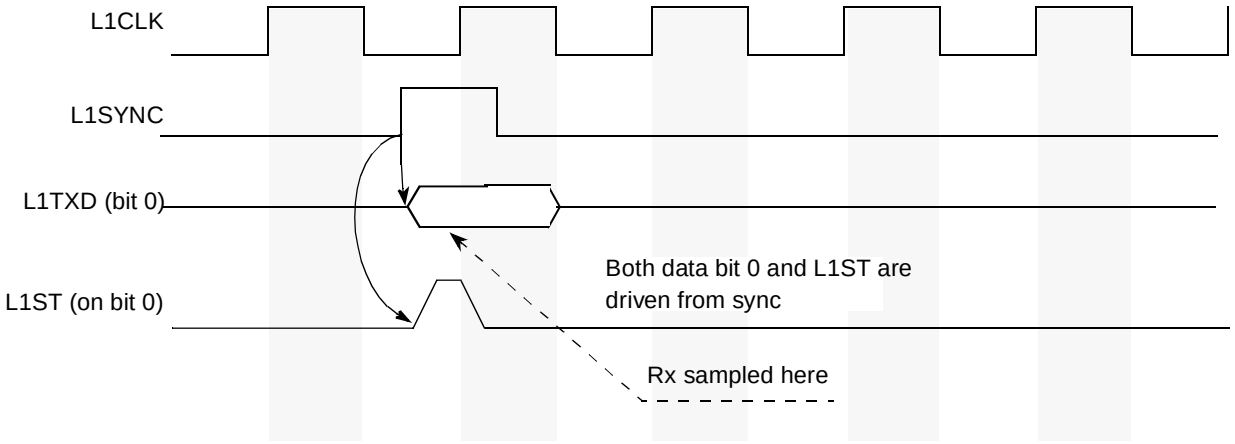


Figure 36-20. Falling Edge Effect when SI*x*MR[CE] =1, SI*x*MR[FE] =1 and SI*x*MR[*n*FSD] = 00 (Sync Starts Before Rising Edge)

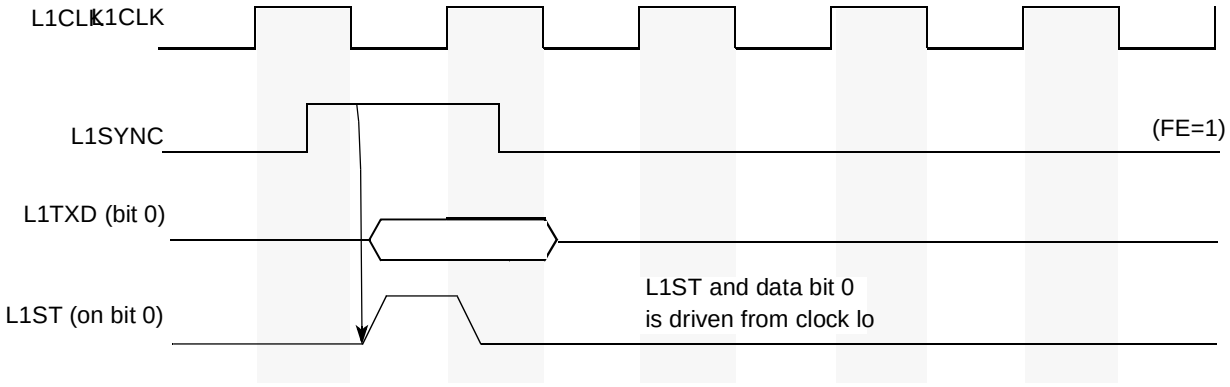


Figure 36-21. Falling Edge Effect when SI*x*MR[CE] =1, SI*x*MR[FE] =1 and SI*x*MR[*n*FSD] = 00 (Sync Starts Before Falling Edge)

The effects of different FE when SI*x*MR[CE] = 0 with no frame sync delay are shown in Figure 36-22 through Figure 36-25.

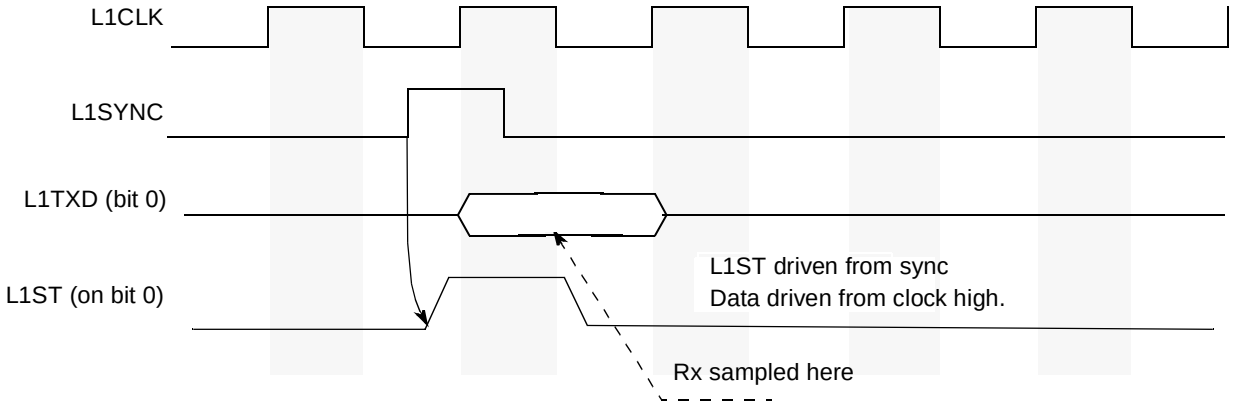


Figure 36-22. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 1 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

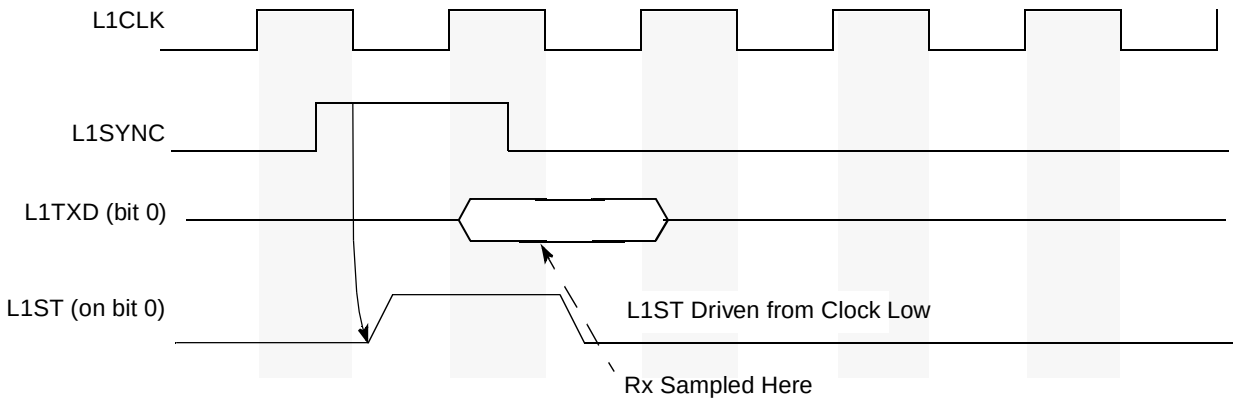


Figure 36-23. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 1 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

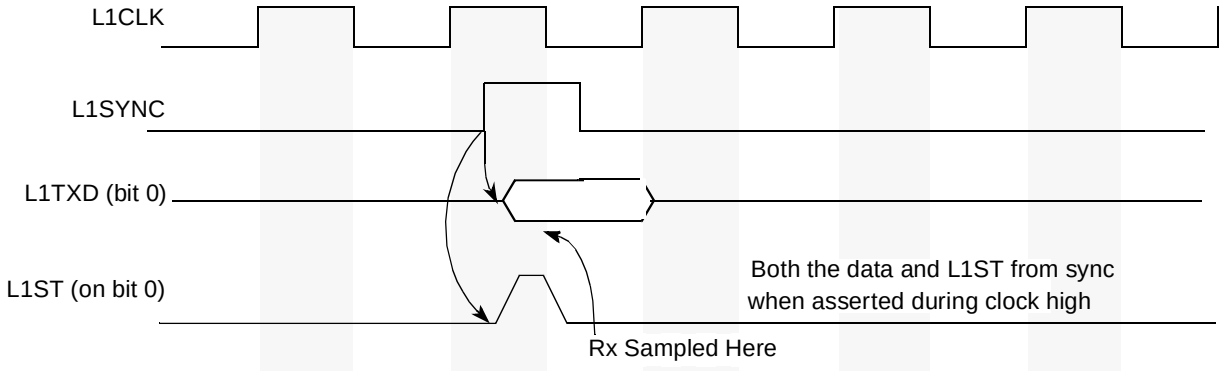


Figure 36-24. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Falling Edge)

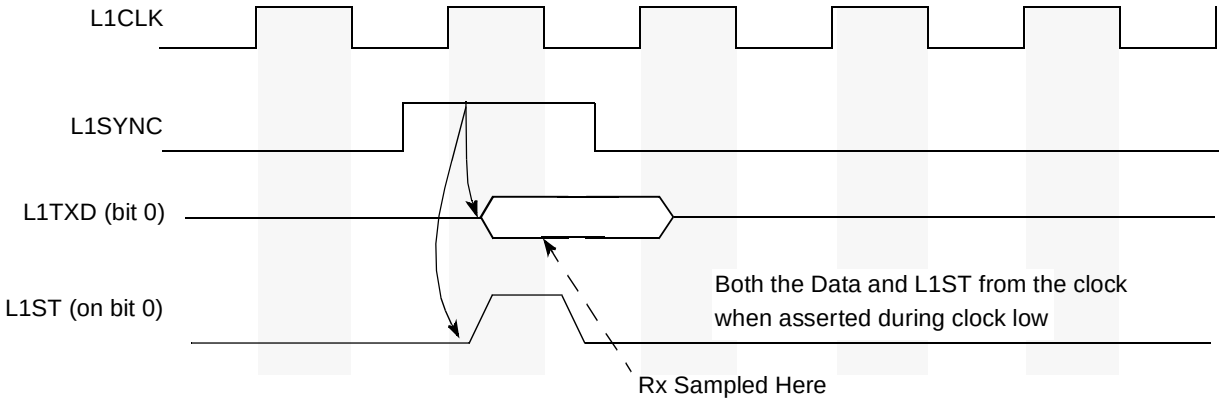


Figure 36-25. Falling Edge Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 0 and SI*x*MR[*n*FSD] = 00 (SYNC Starts Before Rising Edge)

36.6.5 SI RAM Shadow Address Register High (SIRSRH)

SIRSRH, shown in Figure 36-26, defines the starting addresses of the shadow section in the SI RAM for each of the high four TDM channels (TDM A–TDM D).

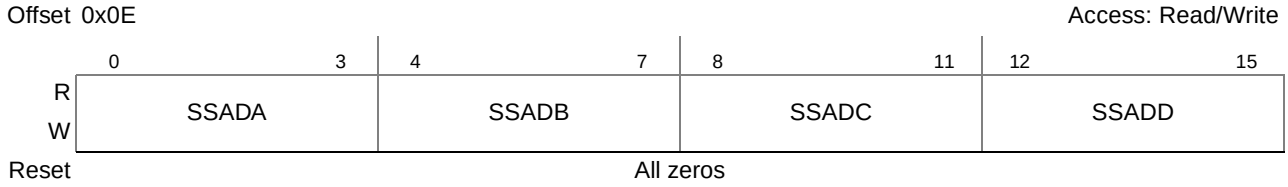


Figure 36-26. SI RAM Shadow Address Register High (SIRSRH)

SI RAM shadow address register field description is shown in Table 36-10.

Table 36-10. 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

36.6.6 SI RAM Shadow Address Register Low (SIRSRL)

The SI RAM shadow address register low (SIRSRL), shown in Figure 36-27. It defines the starting addresses of the shadow section in the SI RAM for each of the low four TDM channels. When the TEG bit in SIMR is set the register has different structure, shown in Figure 36-27.

Offset 0x2E Access: Read/Write

	0–3	4–7	8–11	12–15
R W	SSADE	SSADF	SSADG	SSADH
Reset	All zeros			

Figure 36-27. SI RAM shadow address register low (SIRSRL)

SI RAM shadow address register field description is shown in Table 36-11.

Table 36-11. SI RAM Shadow Address Register Description

Bits	Name	Description
0–3, 4–7, 8–11, 12–15	SSAD*x*	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. 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

36.6.7 SI Command Register High (SICMDRH)

SICMDRH, shown in Figure 36-28, 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 36-28. SI Command Register High

SI Command register field descriptions are shown in Table 36-12.

Table 36-12. 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.

36.6.8 SI Command Register Low (SICMDRL)

SICMDRL, shown in Figure 36-29, 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 0x2A Access: Read/Write

	0	1	2	3	4	5	6	7
R W	CSRRE	CSRTE	CSRRF	CSRTF	CSRRG	CSRTG	CSRRH	CSRTH
Reset	All zeros							

Figure 36-29. SI Command Register Low

SI command register low field descriptions is shown in Table 36-13.

Table 36-13. SI Command Register Low Field Descriptions

Bits	Name	Description
0,2,4,6	CSRR*x*	Change shadow RAM for high four TDM's receiver. 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 CSSRR*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 high four TDM's transmitter. Set CSRT*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 receiver shadow RAM. Cleared as soon as the switch has completed. **Note:** There is a gap between setting the CSSRR*x* bit by the user, and clearing it by the SI. This is because the switch can take place only after frame end.

36.6.9 SI Status Register High (SISTRH)

SISTRH, shown in Figure 36-30, 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 36-30. SI Status Register High (SI*x*STRH)

SI status register high field descriptions are shown in Table 36-14.

Table 36-14. 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.

36.6.10 SI Status Register Low (SISTRL)

SISTRL, shown in Figure 36-31, identifies the current-route RAM, for high four TDM's. SISTR values are valid only when the corresponding SI*x*CMDR bit = 0.

Offset 0x2C — Access: Read Only

	0	1	2	3	4	5	6	7
R W	CRORE	CROTE	CRORF	CROTF	CRORG	CROTG	CRORH	CROTH
Reset	All zeros							

Figure 36-31. SI Status Register Low (SI*x*STRL)

Table 36-15 describes the SI*x*STRL fields.

Table 36-15. SI Status Register Low 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.

36.6.11 SI Multiframe Limits Register (SIML*x*)

SIML*x*, shown in Figure 36-32, holds limit of frame counts for the MCC multiframe. There are two SIML registers for each TDM, one for TDM Rx and the other for Tx.

Offset SIMLA 0x48, SIMLB 0x4C, SIMLC 0x50, SIMLD 0x54, SIMLF 0x5C, SIMLG 0x60, SIMLH 0x64, SIMLE can be accessed by 0x40 or 0x58 — Access: Read/Write

	0	5	6	11	12	13	15
R W	RRC		SRC		ME	—	
Reset	All zeros						

Figure 36-32. SI Multiframe Limits Register (SIMLx)

Table 36-16 describes the SIML*x* fields.

Table 36-16. SIML*x* Field Descriptions

Bits	Name	Description
0–5	RRC	Regular address count. Holds tbe number of repetitions of the frame in current frame. Default value of zero stands for one occurrence of pattern in the current frame.
6–11	SRC	Shadow address count. Holds the number of repetitions of the frame in current frame.Default value of zero stands for one occurrence of pattern in the current frame.

Table 36-16. SIML*x* Field Descriptions

Bits	Name	Description
12	ME	Multiframe enable flag. 0 Multiframe is not enabled. 1 Multiframe mode enabled.
13–15	—	Reserved. should be cleared.

NOTE

In multiframe mode, the last two entries in a frame, should be at least two clocks wide.

36.6.11.1 Multiframe Description

In Multiframe mode, the frame is based on two patterns that repeat several times. The first pattern occurs several times repeatedly, followed by a second pattern that occurs several times repeatedly. The programmer can define two basic patterns in SIRAM. The first pattern is defined in the current RAM, and the second pattern in the shadow RAM. The number of repetitions is defined in the SI Multiframe counter register. The first pattern is configured in the SIML[RRC] field and the second pattern is configured in the SIML[SRC] field. Note that the default value of zero in the SIML[RRC] and SIML[SRC] fields stand for one occurrence of the pattern in Current frame.

Figure 36-33 provides an example of configuring Multiframe mode. The current RAM shows the first pattern, and the shadow RAM shows the second pattern. For this example, the SI Multiframe counter register SIML*x*, is configured to repeat the first pattern three times and repeat the second pattern four times.

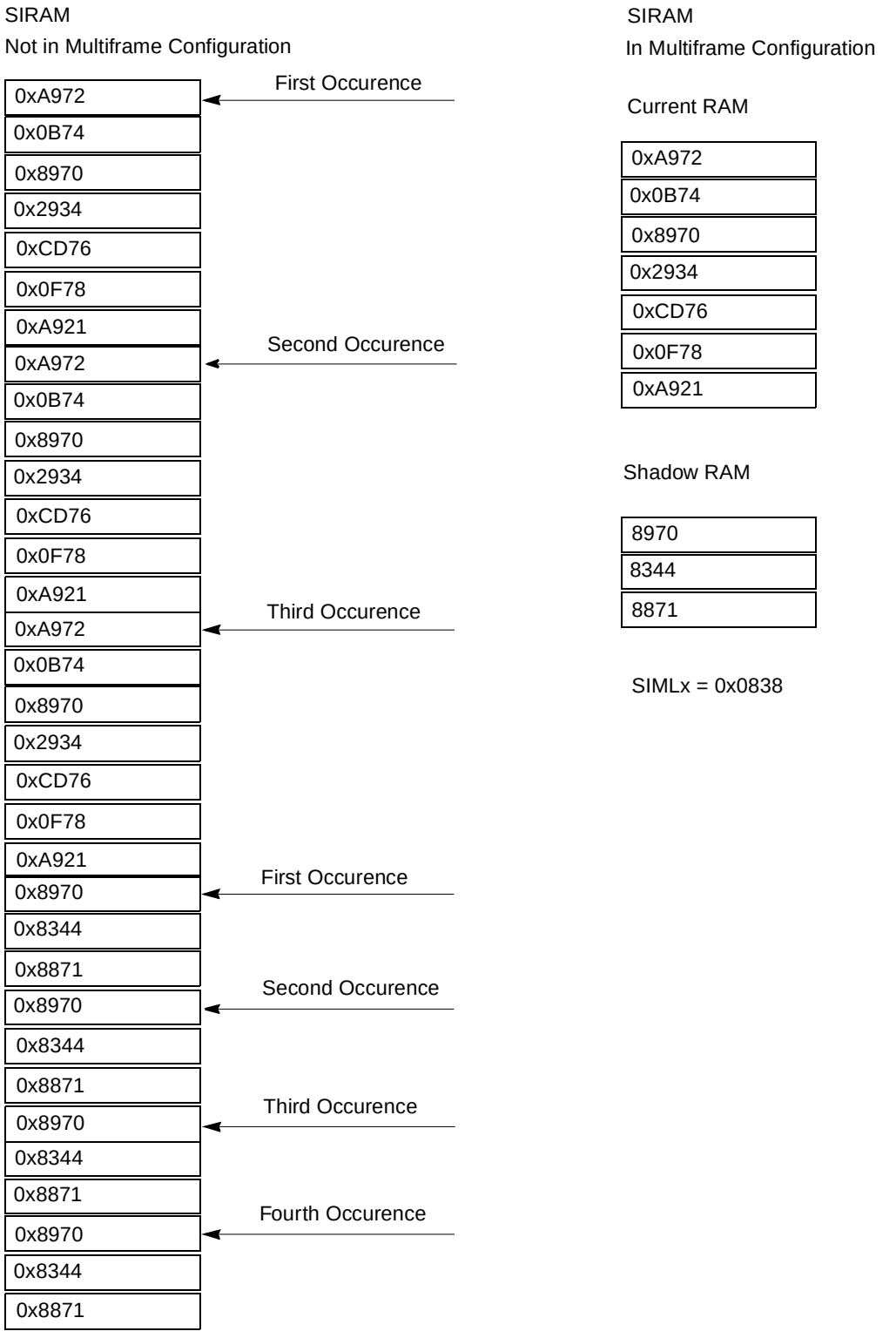


Figure 36-33. Multiframe Example

36.6.12 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 36-34.

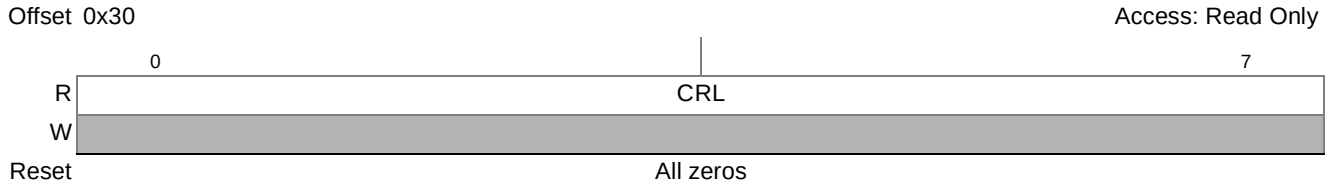


Figure 36-34. SI TDM*x* RAM Counter

Table 36-17 describes the bit fields of SIR*x*RC and SIT*x*RC.

Table 36-17. SIR*x*RC and SIT*x*RC Bit Field Descriptions

Bits	Name	Description
0–7	CRL	Current RAM line number.

36.6.13 SI Enhanced SDM Register (SIENS)

SIENS enhances diagnostic mode for both MCC and 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. When a TSA-related bit is set, SWTR mode is enabled in the MCC command entry, if command entry bit1 is set. The register is shown in Figure 36-35.

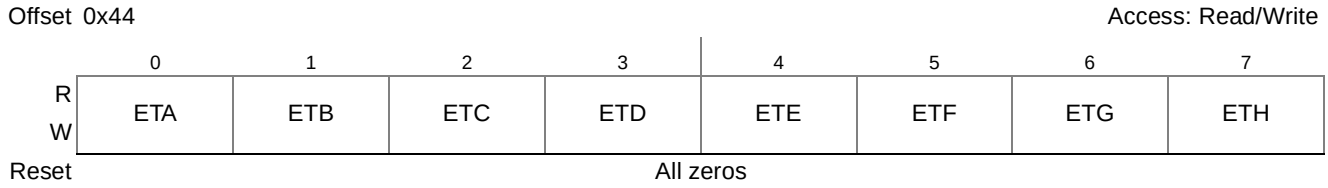


Figure 36-35. SI Enhanced SDM Register

Table 36-18 describes the SIENs bit fields.

Table 36-18. SIENS Bit Field Descriptions

Bits	Name	Description
0–7	ET*x*	0 Normal operation. 1 If bit 1 in MCC entry is set, switch Tx Rx is enabled in this TSA[A–H]. If bit 1 in UCC entry is set, loopback or echo mode is enabled.

36.6.14 SI Speed Register (SISPD)

SISPD, shown in Figure 36-36, holds the speed mode of TDM[A–H].

NOTE

High-speed mode is used under the following conditions:

- Only UCC entries are allowed (bit 0 in each entry must be cleared).

- Each frame must have an even number of entries.

Offset 0x46 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	SPDA	SPDB	SPDC	SPDD	SPDE	SPDF	SPDG	SPDH
Reset	All zeros							

Figure 36-36. SI Speed Register

Table 36-19 describes the SISPD bit fields.

Table 36-19. SISPD Field Descriptions

Bits	Name	Description
0–7	SPD*x*	0 normal operation. The min frequency ratio between the serial clock and the Quick Engine clock is 1:16. 1 High-speed mode.The min frequency ratio between the serial clock and the Quick Engine clock is 1:8 and the max frequency ratio is 1:16. Only UCC entries are allowed.

36.6.15 SI TX Clock Edge Invert (SITXCEI)

SITXCEI, shown in Figure 36-37, inverts the TX clock edge. It is used for high serial clock frequencies.

Offset 0x47 Access: Read/Write

	0	1	2	3	4	5	6	7
R W	TXCEIA	TXCEIB	TXCEIC	TXCEID	TXCEIE	TXCEIF	TXCEIG	TXCEIH
Reset	All zeros							

Figure 36-37. SI TX Clock Edge Invert Register

Table 36-20 describes the SITXCE*x* bit fields.

Table 36-20. SITXCEI Bit Field Descriptions

Bits	Name	Description
0–7	TXCEI*x*	0 Normal operation. 1 TX Clock Edge Inverted. The Tx data is transmitted a serial clock phase earlier then it should be according to SI*x*MR[CE]. For example, If CE = 0, Txd should be driven on the positive edge of the clock, but if this bit is set, it is driven on the negative edge of the serial clock (a serial clock phase before).

NOTE

- TXCEI must be cleared if FSD is cleared.
- The STZ function should not be used when TXCEI = 1.

A case of SI*x*MR[FE] = 1, SI*x*MR[CE] = 0, SI*x*MR[*x*FSD] =01, and TXCEI = 1 is shown in Figure 36-38. L1SYNC is sampled on the rising edge, L1TXD should be driven on the rising edge, and L1RXD should be sampled on the falling edge of the clock. Since TXCEI = 1, L1TXD is driven on the falling edge (a serial clock phase earlier) and L1RXD is still sampled on the next falling edge.

This bit does not affect the strobe timing.

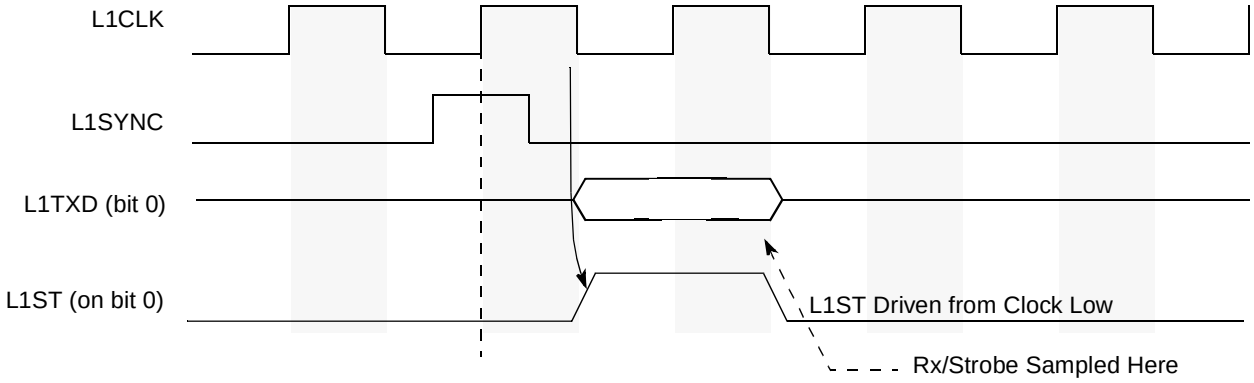


Figure 36-38. TXCEI Effect when SI*x*MR[CE] = 0, SI*x*MR[FE] = 1, and SI*x*MR[*x*FSD] = 01

A case of SI*x*MR[FE] = 1, SI*x*MR[CE] = 1, SI*x*MR[*x*FSD] = 01, and TXCEI = 1 is shown in Figure 36-39. L1SYNC is sampled on the rising edge, L1TXD should be driven on the falling edge, and L1RXD should be sampled on the rising edge of the clock. Because TXCEI = 1, L1TXD is driven on the rising edge (a serial clock phase earlier) and L1RXD is still sampled on the next rising edge. This bit does not effect the strobe timing.

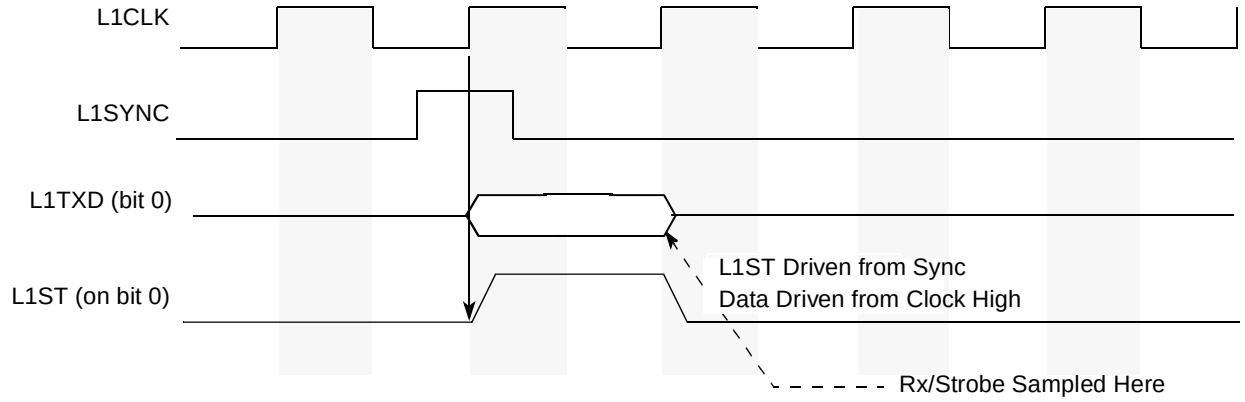


Figure 36-39. TXCEI Effect when SI*x*MR[CE] = 1, SI*x*MR[FE] = 1, and SI*x*MR[*x*FSD] = 01

36.7 SI Functional Description

36.7.1 SI RAM

The SI has a transmit RAM and a receive RAM, each with 512 entries to control TDM channel routing to all UCCs or the MCC. 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 eight 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.

36.7.1.1 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 36-43.

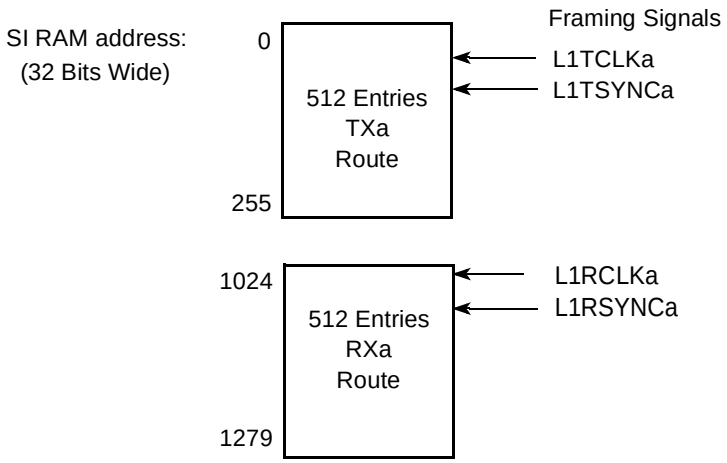


Figure 36-40. One TDM Channel with Static Frames and Independent Rx and Tx Routes

36.7.1.2 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 36-41. 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.

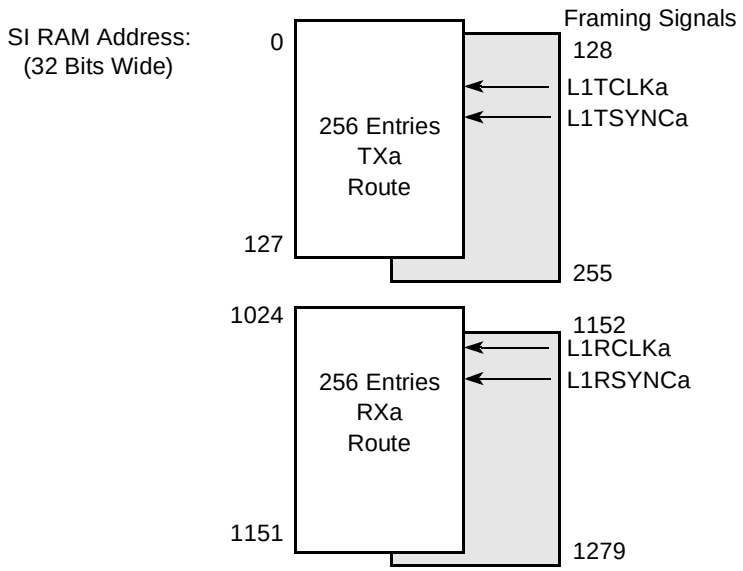


Figure 36-41. One TDM Channel with Shadow RAM for Dynamic Route Change

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.

36.7.1.3 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 reenabled.
- 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 36-42

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 36-42.

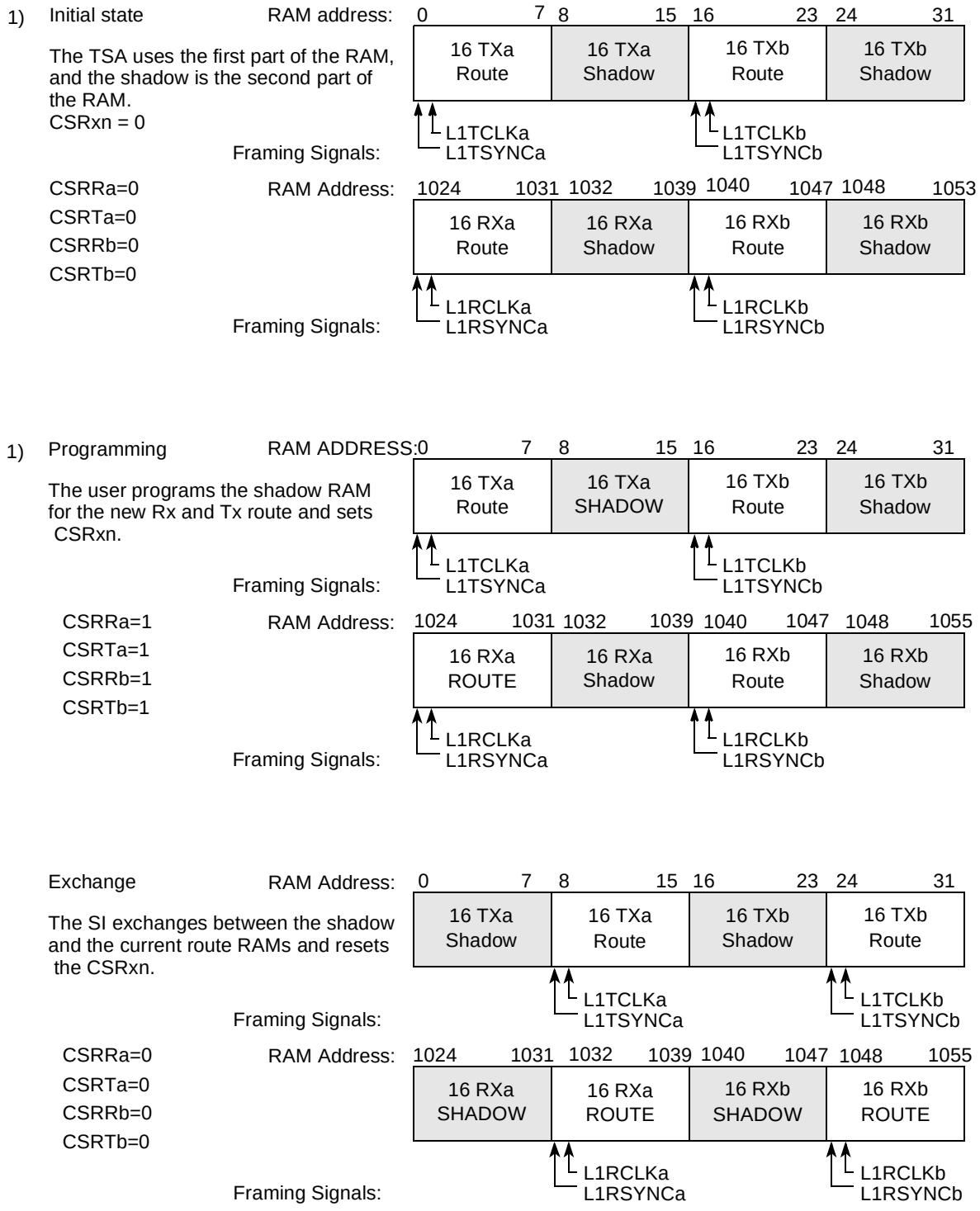


Figure 36-42. Example: SI RAM Dynamic Changes, TDM A and TDM B, Same SI RAM Size

36.7.2 Time-Slot Assigner

The TSA has two separate TSA inside: Rx TSA and Tx TSA.

36.7.2.1 Tx TSA

The Tx TSA implements route selection of data transmitted from the UCCs or MCC 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 MCC or UCC to the TDM This procedure occurs for all TDMs.

36.7.2.2 Rx TSA

The Tx TSA implements route selection of data received from the TDMs to UCCs or MCC 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 MCC or UCC. This procedure occurs for all TDMs.

36.8 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 and MCC.
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.

36.9 SI Application Information

36.9.1 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 36-45. 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 36-21.

.

Table 36-21. RAM Word Descriptions

Entry Number	RAM Word							
	MCC	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
1	0	0	0000	1010	000	1	0	8-bit UCC2
2	0	0	1000	1001	000	0	0	1-bit UCC1 strobe1
3	0	0	0000	0000	000	0	0	1-bit no support
4	0	0	0100	0000	011	0	0	4-bit strobe2
5	0	0	0000	1011	011	0	0	4-bit UCC3
6	0	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.

36.9.2 One Multiplexed Channel with Static Frames

In the example configuration shown in Figure 36-43, 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.

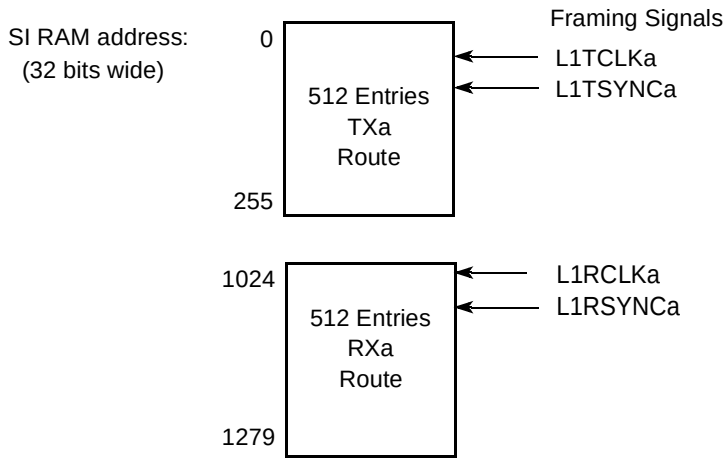


Figure 36-43. One TDM Channel with Static Frames and Independent Rx and Tx Routes

36.9.3 One Multiplexed Channel with Dynamic Frames

A configuration of one multiplexed channel, shown in Figure 36-41, 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 36-44.

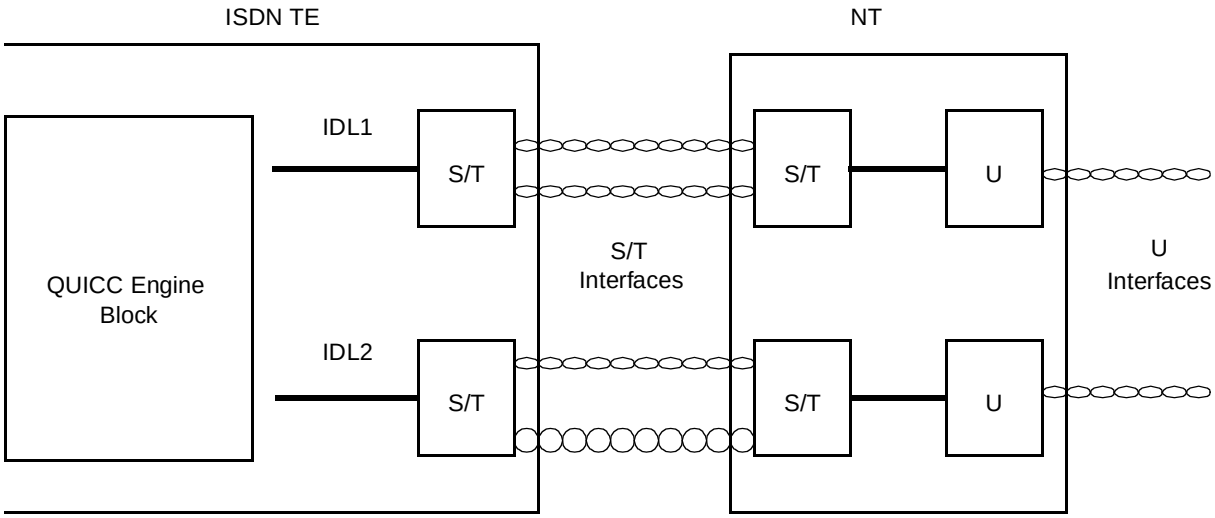


Figure 36-44. Dual IDL Bus Application Example

36.9.4 IDL Interface Example

An example of the IDL application is the ISDN terminal adaptor illustrated in Figure 36-45. 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 36-22.

Table 36-22. IDL Signal Descriptions

Signal	Description
L1RCLKx	IDL clock; input to the device.
L1RSYNCx	IDL sync signal; input to the devce. 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:H 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 36-45.

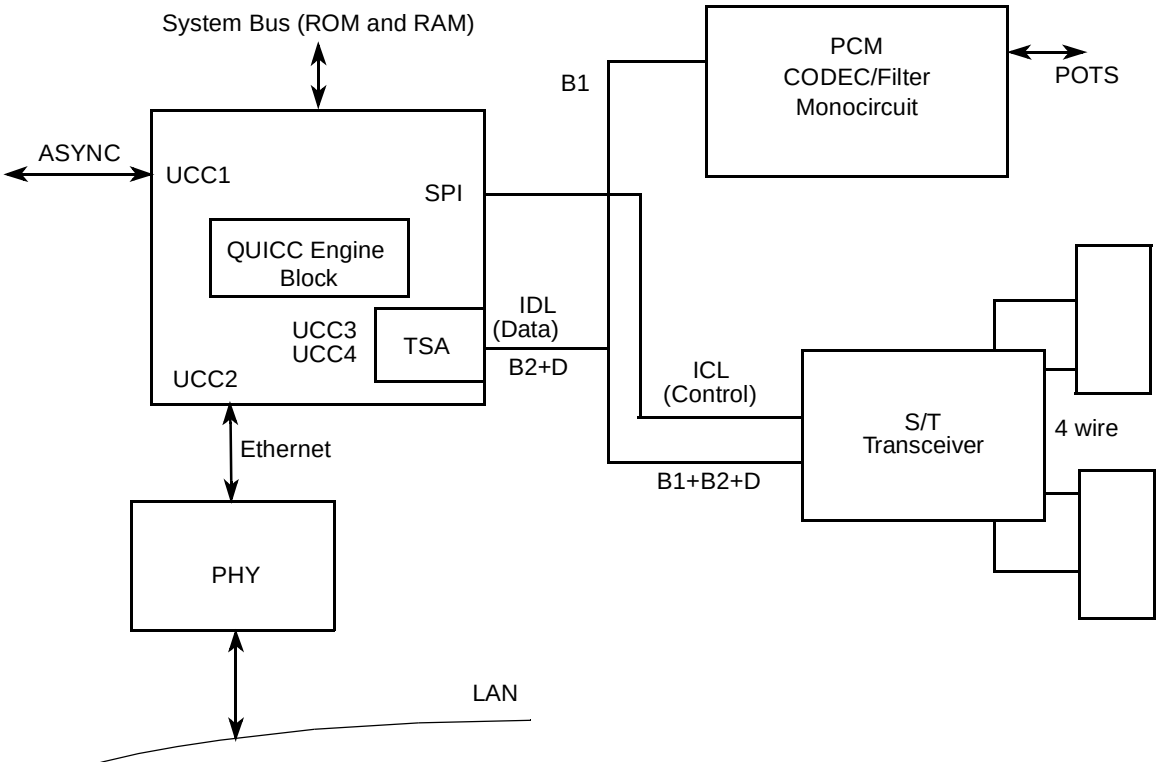
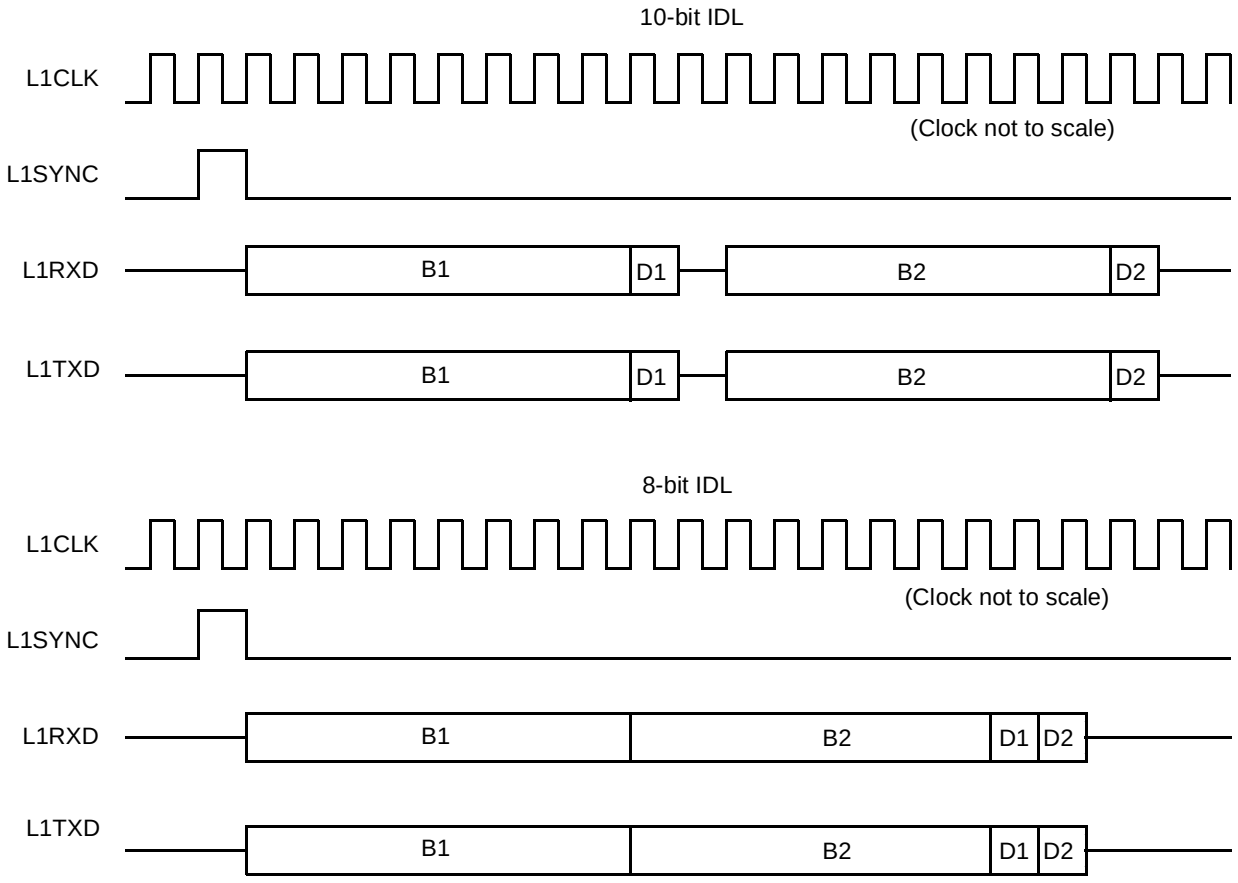


Figure 36-45. IDL Terminal Adaptor



NOTE: L1RQn and L1GRn are not shown.

Figure 36-46. 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.

36.9.5 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 36.9.1, “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 37
Point-to-Point Protocol (PPP)

37.1 Overview

The QUICC Engine implementation of the PPP is compliant with the RFCs 1661, 1662, 1990, 2686, and 3153. This chapter describes the functionality and data structures of the PPP protocol, the Multilink-Multiclass PPP (ML-MC) protocol, and the PPP mux protocol.

The Multilink (ML) protocol maximizes the bandwidth for real time multimedia flows over low-bit rate links, while minimizing the end-to-end delay. The ML protocol provides packets to be split into fragments that run over several TDM links that are part of a bundle. As shown in Figure 37-1, each TDM link can be configured as a single link running the PPP protocol or configured as part of a bundle running the ML-MC protocol.

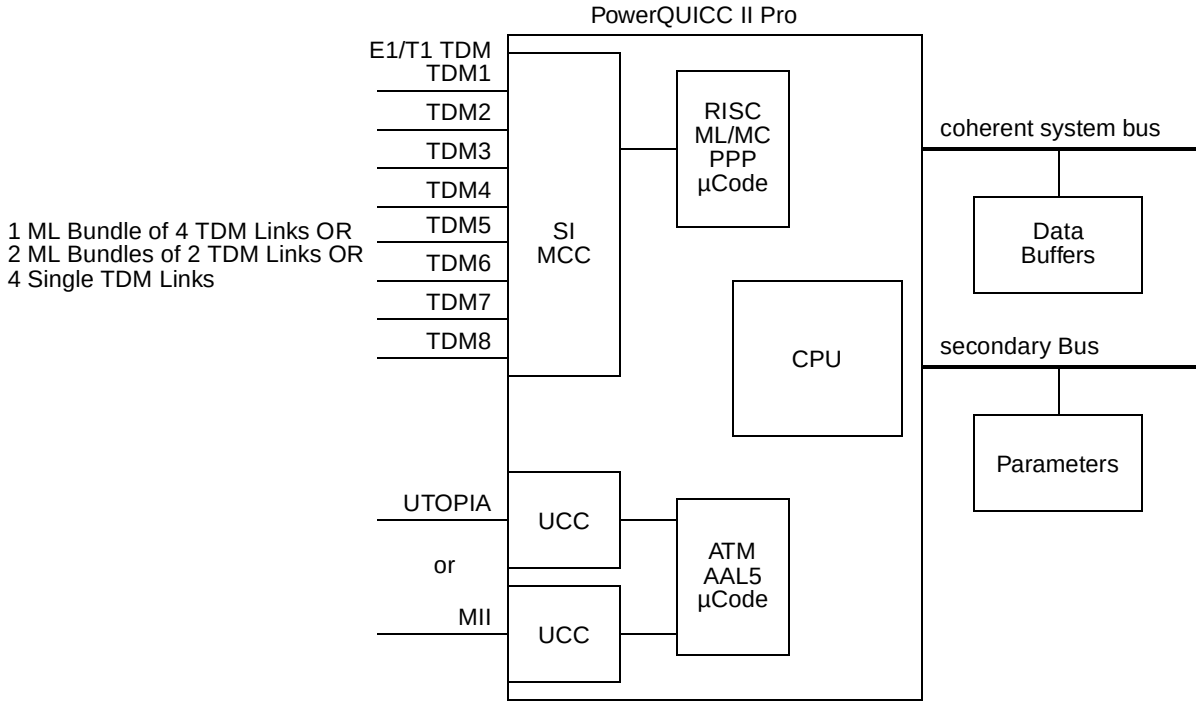


Figure 37-1. Example ML-MC PPP System

37.2 Terminology

This section defines the terms used in this document. When applicable, the RFC number in which the term is used is specified.

Figure 37-2 shows the frame structures that can be processed by the PPP/MLMC/PPP mux protocol stack.

Plain PPP Frame Received on a Link

Address & Control	PID	Information (Up to 1500 Bytes)	FCS

PPP MUX Superframe Received on a Link

Address & Control	PPP MUX	PPP Subframe Information	PPP Subframe Information	PPP Subframe Information	PPP Subframe Information	FCS

ML/MC PPP Fragment Received on a Link

Address & Control	ML/MC Header	PID	Information	FCS

PPP MUX Superframe Over ML/MC Received on a Link

Address & Control	ML/MC Header	PPP MUX	PPP Subframe Information	PPP Subframe Information	PPP Subframe Information	FCS

Figure 37-2. Typical Frames Handled by the RISC

Table 37-1. Terminology

Term	Definition
Frame	The unit of transmission at the data link layer (RFC 1661).
Packet	The basic unit of encapsulation, which is passed across the interface between the Network Layer and the Data Link layer (RFC1661). **Note:** The terms “packet” and “frame” are used in this document interchangeably.
MultiLink (ML) Fragment	A portion of a Network Layer packet sent as a frame over a link belonging to a MultiLink bundle. The MultiLink bundle is a virtual PPP link-layer entity (RFC1990).
Maximum Receive Unit (MRU)	The maximum size of a ML Fragment on a given Link.
Maximum Received Reconstructed Unit (MRRU)	The maximum number of octets in the Information fields of reassembled packets.
Subframe	A PPP encapsulated frame when concatenated into a single PPP multiplexed frame (RFC3153).
PPP Mux super frame	A frame which contains multiple PPP Subframes.
Link queue	An ingress or egress queue which is directly associated with a link and does not belong to the ML/MC traffic. This queue is for Plain PPP frames. It could also be a PPP Mux frame with no ML/MC header.
Queue Descriptor	Data structure which has the management parameters of a queue of BDs. Each BD ring is managed by such a structure, both, for egress and ingress traffic.

37.3 Features

- Conformance to RFC 1661—Point-to-Point Protocol (PPP)
  - PFC protocol field compression mode—Dynamically tested
  - PID is copied to BD
- Conformance to RFC 1662—PPP in HDLC like Framing
  - Receives both kinds of interframe fill: contiguous 1 and flag sequence
  - User-defined FCS: either 16 bit FCS or 32 bits FCS
  - Programmable address and control characters—The user may program one ACC in addition to 0xFF03
  - Address and control field compression (ACFC) is supported
  - Programmable MRU (maximum received unit)
  - No support for Shared Zero Mode
- Conformance to RFC 1990—The Multilink Protocol (MP)
  - User-defined sequence number: either 12 bit or 24 bit
  - Fragment loss detection
    - User-programmable fragment loss window size
    - Frame is discarded in detection of lost fragment
    - Fragment lost counter
  - Programmable MRRU
  - No support of self-describing padding
  - No support of prefix elision
- Dynamic addition and removal of link in a bundle
  - Addition of link.
  - Supports dynamic removal of link from a bundle.
- Conformance to RFC2686 MultiClass extension to Multilink PPP
  - Supports up to 8 classes
  - User-defined class field: either 2 CLS bit or 4CLS bit
  - Frames are enqueues in separate queues by class
- LCP frames handling
  - LCP frames from all links are enqueued in the dedicated LCP Link core queue which uses a dedicated BD ring
  - LCP frames received as ML-MC frames are enqueued in the class core queue
- Free buffer pools (FBPs)
  - Single FBP on ingress
  - Multiple buffer pools on egress
- Supports channelized MCC and PPP
- Statistics

  - The following statistical information per link is provided:
    - FCS error counter
    - Invalid address and control characters counter
    - Aborted fragments/frames counter
    - Illegal fragments/frames counter
    - Transmit and receive plain PPP frames counter
    - Transmit and receive MLMC PPP fragments on this link
    - Transmit and receive byte count per link (32 bit counter)
    - Transmit idle counter
    - MRU exceeded counter
    - Link queue busy counter
  - The following statistical information per class is provided:
    - Fragment loss counter
    - Window full counter
    - Class queue busy counter
    - Demultiplexer queue busy counter
    - MRRU exceeded counter
    - Receive frames counter
- Connectivity to up to 8 T1/E1 TDM interfaces
- Conformance to RFC 3153, PPP MUX handling
  - Supports up to 256 Sub-Frames per fragment
- Decapsulating received PPP Mux frames into sub frames, extracting PID information automatically
- Support aging discard for PPP Subframes
- Support time-out for stopping encapsulation process in PPP MUX frames
- Support up to eight queues per link
- Weighted fair queuing algorithm for class and queue selection
- Automatic removal of empty queue from queue selection process
- LCP transmission
- Packet insertion on a link when working in ML/MC mode
- No data copying

37.4 Operation

37.4.1 Receive (Rx)

When a link begins to receive a fragment, the received data is stored in an internal memory buffer pointed to by Rx_IB_ptr within the Link Parameter table. After a sufficient amount of data is received, the microcode begins parsing the content of the frame.

The address and control fields are checked, and then the PID is compared. There are four possibilities: MLMC-PPP fragment, plain PPP frame, PPP MUX frame and LCP frames. The receiving process continues until the whole frame/fragment is received and checked for correct FCS. After FCS verification, the frame is enqueued into the appropriate queue.

In the case of plain PPP traffic, the Rx process enqueues the received frame to a BD ring for handling by the Host. It is referred to as a Link queue. In the case of ML/MC, the Rx receives on each link of the bundle a fragment and re-assembles all the fragments of the same frame. After reassembling, the Rx enqueues the complete frame to a BD ring for handling by the host. Every fragment is received in a separated buffer. This is referred to as ML/MC queue. The Rx detects and supports fragment loss according to RFC 1990.

The following types of packets are discarded. A counter for each type of error is updated:

- Packets with address and control characters that do not match the characters negotiated for the link
- Packets that are too short
- Packets with a bit stuffing error
- Packets with bad FCS.

37.4.1.1 Reception of Plain PPP, LCP and PPP MUX Frames on a Single Link

There are three receive queues for each link. These queue frames do not have the PPP ML MC header. Each of the three queues is dedicated for a specific type of frame: one is for plain PPP frames, one for PPP mux frames, and one for LCP frames. The host has two methods of handling data buffer pointers. One is to use a Free Buffer Pool (FBP) where buffer pointers are fetched from the FBP. In this case all the BDs in a queue do not contain a valid pointer upon initialization. The other option is to have an initialized BD which contains a data pointer. The RISC will swap the pointers between a BD and the buffer currently used by the receiver. Only for the first queue (plain PPP frames), the host can select which of the methods is being used.

For the LCP queue the host prepares a BD Ring where the BDs do contain data buffers pointers. The RISC swaps the pointers in the BD with the one currently used for reception of the frame.

Plain PPP frames are enqueued into the plain PPP queue and PPP mux frames are forwarded to the PPP demultiplexing (Demux) process described later.

37.4.1.2 Reception of Multi Link-Multi-Class PPP Fragment

In ML/MC-PPP frames, the RISC expects two kinds of packets: data packets and PPP mux packets. The reconstruction of a complete frame from its fragments is done in two stages: the front-end and back-end. The front-end process collects the fragments of each class, detects a completion of a frame or a fragment loss, and then forwards each completed frame into the back-end process—the packet reconstruction task—for enqueuing into the appropriate queue.

Each fragment is stored in the queue only if it was completely received with the correct FCS and no other violation of the setup. After a successful reception of a complete frame, the back-end parses the frame and checks if it's a PPP mux frame or not according to the PID. In the case of PPP mux, the receiver enqueues the frame to the demultiplexer; otherwise the frame is enqueued to a BD ring for handling by the host.

37.4.1.3 Reception of PPP Mux Frame

The demultiplexer (demux) is a stand alone centralized process which handles PPP mux frames. It takes a received frame, from the back-end of the receiver, analyzes its content and de-multiplexes a PPP mux frame by assigning each of its encapsulated subframes to a host queue.

37.4.2 Transmit (Tx)

The transmitter also has also two separate sources for transmission. One is transmission from a Link queue of plain PPP frame or a PPP MUX frame, and the other is transmission of a ML/MC fragment from a bundle.

For each of these sources there is a dedicated process for building and encapsulating a valid frame.

37.4.2.1 Transmission of Plain PPP, LCP, and PPP MUX Frames on a Single Link

Each link has an associated queue structure that supports transmission of plain PPP frames (typically LCP frames) and PPP MUX frames. The Link queue has a higher priority than the ML MC queue of a bundle.

There is an option to define up to eight queues that are related to each link. Selection of a queue is done by the weighted Fair Algorithm (WFQ).

37.4.2.2 Transmission of ML/MC PPP

This process is divided, as on the receiver, to two stages: the front-end and back-end.

The front-end is responsible for transmission and HDLC framing of fragments which are prepared by the back-End process. The back-end is responsible for scheduling among classes if applicable, and preparing valid fragments for transmission.

The microcode supports up to eight different classes per a ML/MC bundle. Selection of each class for transmission is also done by a WFQ algorithm. The class field (CLS field) for the ML/MC header is taken from a Class Parameter Table, as is the incremental sequence number.

Each transmit class contains only one QD under it, which can be of type ML-MC MUX or ML MC non MUX.

37.4.2.3 Transmission of PPP MUX

The MUX encapsulation is a stand alone centralized process which generates PPP mux super-frames. Each bundle could have one of its associated classes assigned as a MUX queue. The process is an upper layer in the PPP transmission. Once a superframe is built, it is handled by the back-end of the transmitter for fragmentation if this is a Multi-Link queue, or by the link itself if this is a Link queue. For PPP mux frame the procedure for encapsulating subframes into a PPP mux frame is stopped if one of the following happens:

- The next subframe to be added to the frame will cause the total length to exceed the maximum value allowed for a super-frame. (This maximum value is programmable.)
- A time-out timer has elapsed.

Each sub-frame could have a time stamp and an aging mechanism timer is defined in order to ensure that “older” sub frames are not transmitted. When the encapsulation process starts building a new superframe, the time stamp of each BD is examined. If the difference between the current time and the time stamp has exceed the allowed time the sub-frame is discarded and will not be queued into the superframe. A counter increments for each subframe discarded due to aging.

The code fully complies to RFC 3153 with respect to PFF and LXT handling. PFC is also supported.

37.4.3 Supporting Channelized MCC and PPP

Running channelized MCC and PPP configurations is possible. The PPP microcode data structure is combined with the MCC data structure as follows:

- The Link Parameter Table of each PPP link has the size of 4 MCC channel specific parameter RAMs. The user must reserve 4 contiguous channel specific parameter RAM for each PPP link.
- The PPP Global Parameter RAM is placed in the MCC page in the location MCCbase+0x90.
- The rest of the parameter tables can be located in MURAM, as long as they do not overlap MCC channelized resources (i.e the channel specific parameter RAM and the channel extra parameter RAM).
- The receiver distinguishes between a MCC channel and a PPP link according to the programming of RSTATE. The user should initialize the RSTATE/LRSTATE to 0x HH80_0000 for non MLMC PPP MCC channel. For ML MC PPP operation a dedicated host command will initialize RSTATE/LRSTATE to the appropriate value.

37.5 Data Structures

The following sections briefly describe the data structures defined by this package. A global parameter table contains general parameters for the MCC operation, under which this PPP microcode package runs, and matches the existing MCC global parameter page. This page has been expanded for the operation of this package.

In addition to the global page expansion, other dedicated parameter tables have been added for ML/MC support. These additional tables are described in following sections.

37.5.1 Link Parameter Table

The Link Parameter Table (LPT) contains parameters that are link-related. The location of this table is determined by mapping of a TDM link by the serial interface into a specific MCC channel. The size of this table is larger than the MCC channel specific parameters. Therefore, this table consumes several traditional MCC channels. The LPT is accessed by the RISC on each receive and transmit request, and the initial frame/fragment parsing is done on the link level. The LPT assigns each link with a number and associates each link to a bundle parameter table (BPT). The LPT also contains various link level statistics and management parameters for the receive queues.

37.5.2 Bundle Parameter Table

The Bundle Parameter Table (BPT) contains pointers and parameters that are valid for a bundle: Free Buffer Pool Parameter Table pointer, Class management and tables, Bundle mode bits.

NOTE

If a link is not a member of any bundle, a Bundle Parameter Table still has to be initialized. In such a case the Bundle Parameter Table will be smaller. The purpose for this is to remove redundant parameters on a link and a bundle level and to avoid duplication of parameters in a multi-link setup.

37.5.3 Class Parameter Table (CPT)

The class parameter table (CPT) manages the reconstruction of fragments and the fragment loss using a buffering queue. The buffering queue is represented by an external ring of BDs (WBD ring) and an internal status ring. After reconstruction, the ML/MC PPP frames are stored in a queue, managed by the CPT. The CPT also contains various class level statistics.

Each queue is managed by a queue descriptor. The user initializes a BD ring, leaving the data pointer of each BD blank. A free buffer pool is used to manage the memory allocations of the system.

37.5.4 Free Buffer Pools

The transmitter is capable of handling selection of Tx buffers from four pools. This is useful when the traffic mix contains small sub-frames and bigger data frames. The application designates to each type it's own pool and the transmitter returns the buffer to the specified pool once transmission is done.

The receiver interacts with one buffer pool. This size of the buffers is of the size of MRU in the ML/MC setup.

Note that LCP frames are received into a BD ring where BDs contain data pointers, so reception of LCP frames does not interact with the Free Buffer Pool.

37.5.5 Weighted Fair Queue (WFQ) Table

This is a structure which controls the selection of one queue from up to eight queues on a link or one class of up to eight classes. In a situation where the application requires such differentiation the user defines this structure for each set of queues. Priorities can be set as WFQ, Round Robin, or fixed priority.

37.5.6 Queue Descriptor (QD)

This data structure is associated directly with a queue and contains all the parameters managing it. It points to a buffer descriptor ring (the queue) and defines the mode of operation for this queue. There are a few modes of operation and each queue has its attributes. It is a resource which is shared between the RISC and the host.

37.5.7 Data Structures for RISC Usage

On top of the above data structures there are data structures which are for RISC usage. Some do not have to be initialized and only the memory space should be allocated. Fragment descriptors which are used by the transmitter back-end process is one example for this type of data structures. On the receiver side, a window BD (WBD) is an example for such a data structure. They are described in detail later.

Figure 37-3 and Figure 37-4 illustrate all the data structures described above.

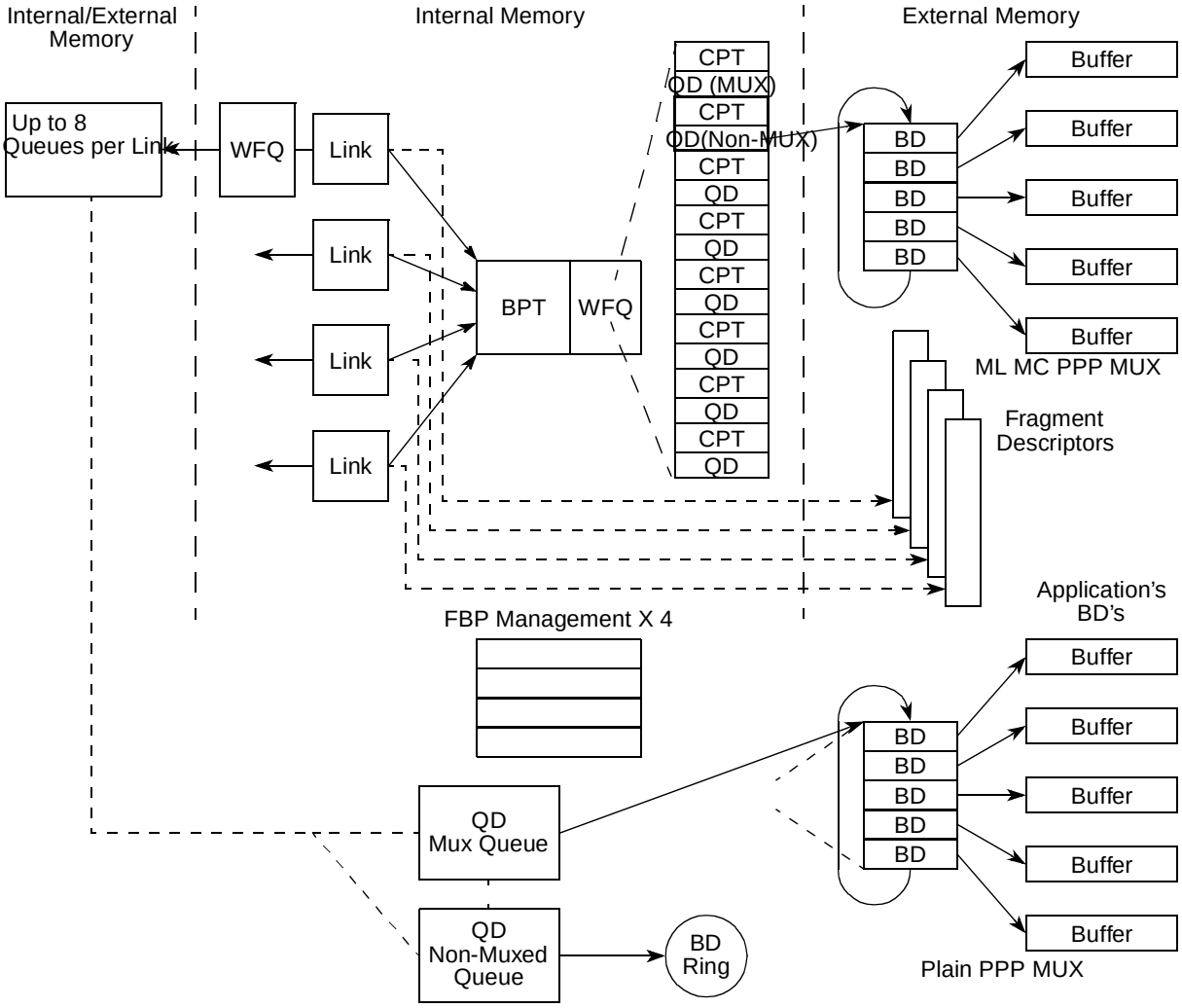


Figure 37-3. PPP Tx Parameter Structures

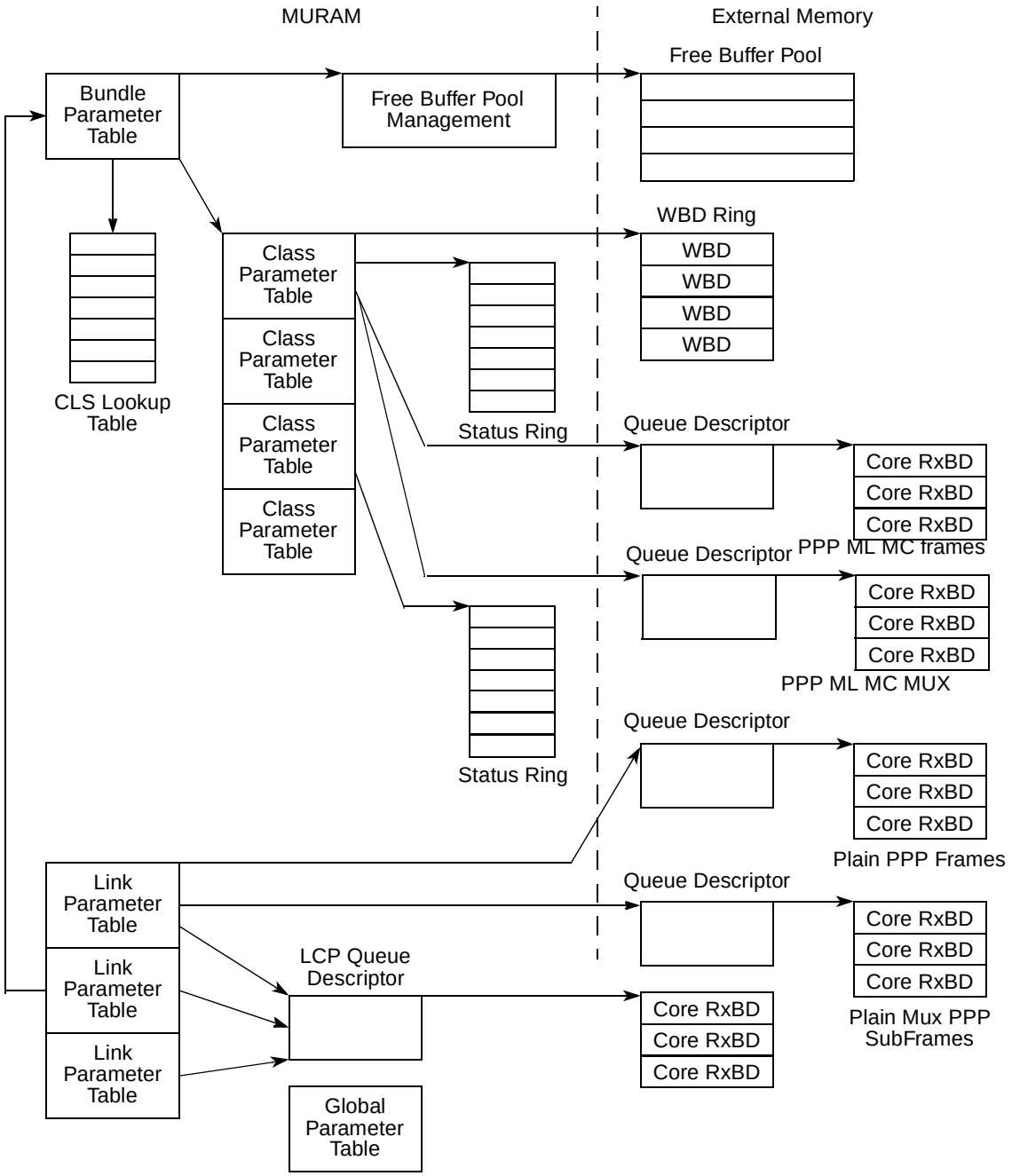


Figure 37-4. PPP Rx Parameters Structures

37.6 Receive Process

37.6.1 Address and Control Characters

This section describes the handling of address and control characters. The default ACC characters are 0xFF03. The user may negotiate other values for each link. The default address and control characters are always recognized.

The user may program other values using the RxACC field in LPT (refer to Section 37.11, “Link Parameter Table (LPT)”). When the user wishes to use only the default address and control characters, initialize the ACC field to 0xFF03. This user programmable ACC is compared when a packet is received. The received ACC is first compared to default Address and Control characters, and then to the ACC field.

When the first two received bytes do not match the default or the ACC programmed value, and RxACFC bit it set, the frame is regarded as a frame with compressed ACC, otherwise the frame is regarded as containing unrecognized address and control characters. Frames containing unrecognized address and control characters are discarded and the counter Rx_IACCNT increments.

The following figure describes the address and control handling on receive side.

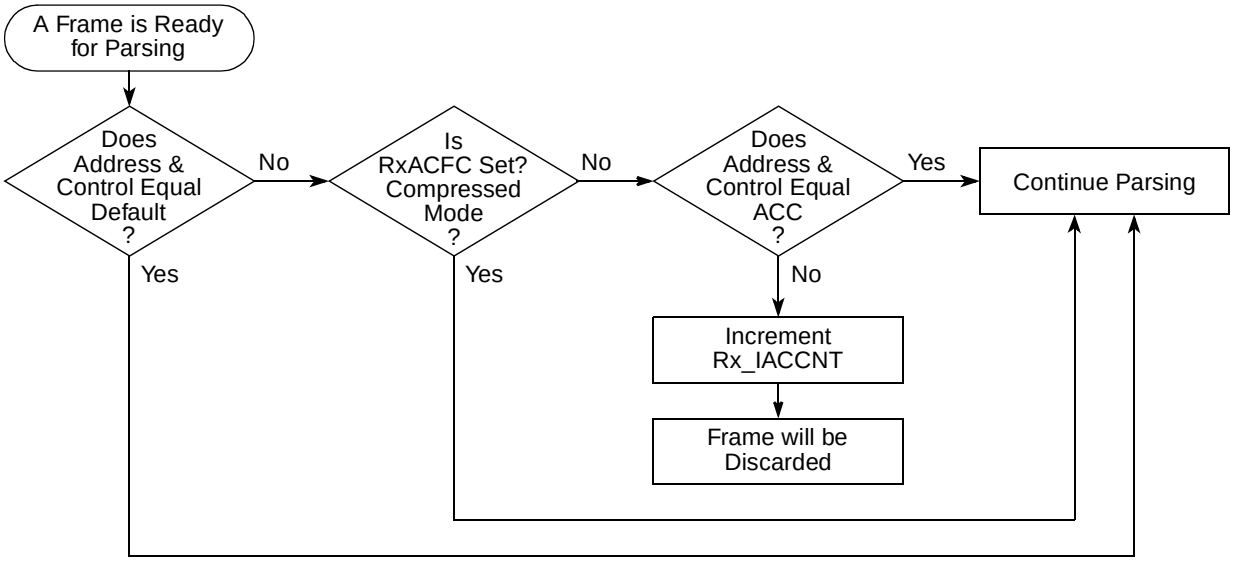


Figure 37-5. Rx ACC Flow

37.6.2 PID on Receive

The PID may or may not be compressed. The receiver handles this dynamically. The receiver checks the LSB bit of the first byte of the PID field. If it is set, the PID is regarded as compressed and it is decompressed for further checks by adding 0x00 as the MSB. If it is cleared the PID is regarded as non-compressed, and all further checks are done on the 2-byte PID field. The PID is not treated as part of the data and is being isolated and put in the BD for host handling. The receiver recognizes several types of PID and acts accordingly. Recognized PIDs include: plain PPP, PPP mux, ML/MC PPP and LCP. The following sections discuss the receive processing for each of the packet types.

37.6.3 Receive of Plain PPP Frames

Plain PPP frames are placed into Link queues. The Link queue is described by a queue descriptor (QD) and a BD ring. Both are initialized by the host. If working in Free Buffer Pool mode enabled (QD[FBP]=1), each BDs in the Link RX BD ring is prepared without any data pointer. The Link Parameter Table includes a valid data pointer from the ML/MC Free Buffer Pool. All Non LCP packets are using the FBP (which has smaller buffers size). The buffers are filled and once the FCS is checked and there are no other errors during reception, the Rx retrieves a free BD from the Link queue, updates its status bits (i.e E, PID), length and inserts the pointer into the BD. Afterwards, new pointer is taken from the pool for the next frame on the link.

If working in Non-Free Buffer Pool mode, the BD ring is initialized with valid data pointers and the RISC swaps the pointer from the BD with the pointer currently used for receiving the frame, and which is also initialized by the application.

37.6.4 Receive of LCP Frames

Once the address and control characters are checked and the PID is identified as a LCP packet the packet is routed to a dedicated queue common to all the links in the bundle if applicable.

This queue is also described by a queue descriptor (QD) and a BD ring. Both are initialized by the host. The BDs are smaller than the plain PPP BDs and each BD in the Link Rx LCP BD ring is initialized with a data pointer. The Link Parameter Table has a dedicated data pointer for LCP packets. All LCP packets use buffers which could have larger size to accommodate default 1500 bytes packets. Once the buffer is filled and after the FCS is checked and there are no other errors during reception, the Rx retrieves a free BD from LCP queue and updates its status bits (i.e E, PID) and swaps the pointer currently in the Link with the one in the BD. The pointer taken from the BD is used for the next LCP packet reception.

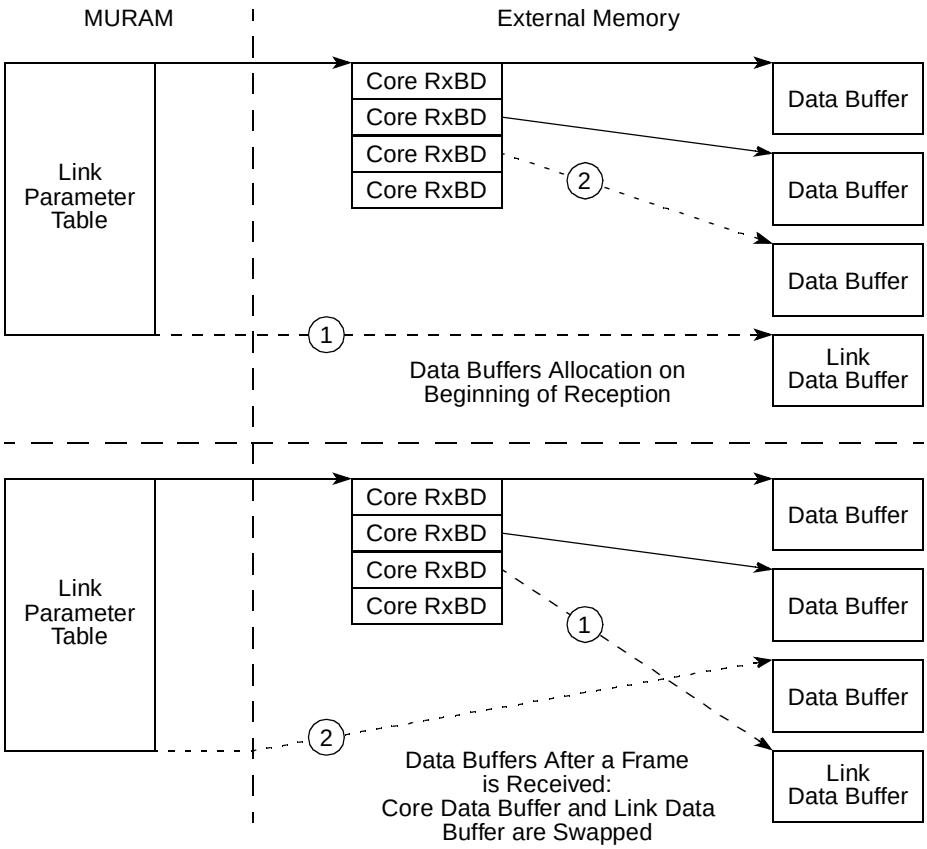


Figure 37-6. LCP Buffers Handling

37.6.5 Reception of ML/ MC PPP Fragments

Reception of ML/MC traffic is done in two independent processes—the front-end and the back-end. Once the front-end has finished processing a PPP packet it activates the back-end for further processing and interfacing to the Host. The front-end uses data structures prepared by the host but are completely managed by the RISC. Those structures and the flow of the process are described later. See Section 37.6.5.1, "WBD and Status Ring."

The receiver fills a data buffer pointed by a valid pointer from the FBP which resides in its table. Once the FCS is checked and no other errors are found, the receiver updates the appropriate WBD in external memory along with its associated status entry in the internal MURAM memory. Location in the WBD is determined by the sequence number of each fragment which is maintained by the receiver on a per class basis. Each WBD has a status nibble in the internal memory. The Valid bit is set in the appropriate status nibble. If this fragment is the beginning of frame, the B bit will be set. In case of an ending fragment, the E bit is set. The WBD is updated with the length of fragment and with the inner PID in case of a beginning fragment. If the Full bit (which is updated automatically by the QUICC Engine block: set upon an error in the fragment) is not set in the Status Ring, the RX writes the data buffer pointer to the WBD and brings a new data buffer from the Free Buffer Pool. Otherwise, the RX swaps data pointers—the data buffer pointer (which points to the received data) is copied to WBD and the data pointer currently in the WBD is copied

to the Link Parameter Table as the next Rx data buffer pointer. In FBP mode the host has to return each processed BD pointer back to the FBP.

Another mode of operation for the QUICC Engine block is not using the FBP- swap mode. Instead the host has to initialize the BDs in the BD ring with valid data pointers. The WBDs should also be filled with valid data pointers and the status ring for each class should be initialized with the Full (F) bits set. In this mode the host swaps the data pointers in the BD ring with new pointers on each receive operation. The QUICC Engine block will swap pointers between the BD ring the WBD ring and the link pointer dynamically.

When a complete frame is found (see Section 37.6.5.2, “Complete Frames Handling”) it is enqueued to a Class Queue. Each BD is updated with the buffers status, fragment’s length and the data buffer pointer. After the Host processes the fragments, the Host clears the empty bit in the BD and returns the pointer to the FBP.

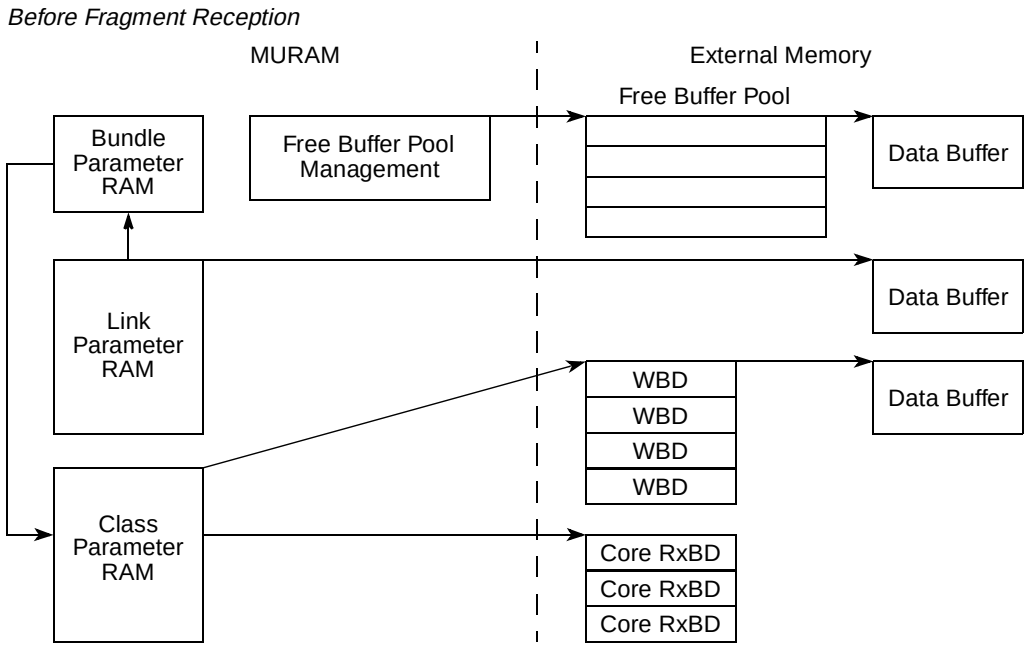


Figure 37-7. Data Buffer Pointers State Before Fragment Reception

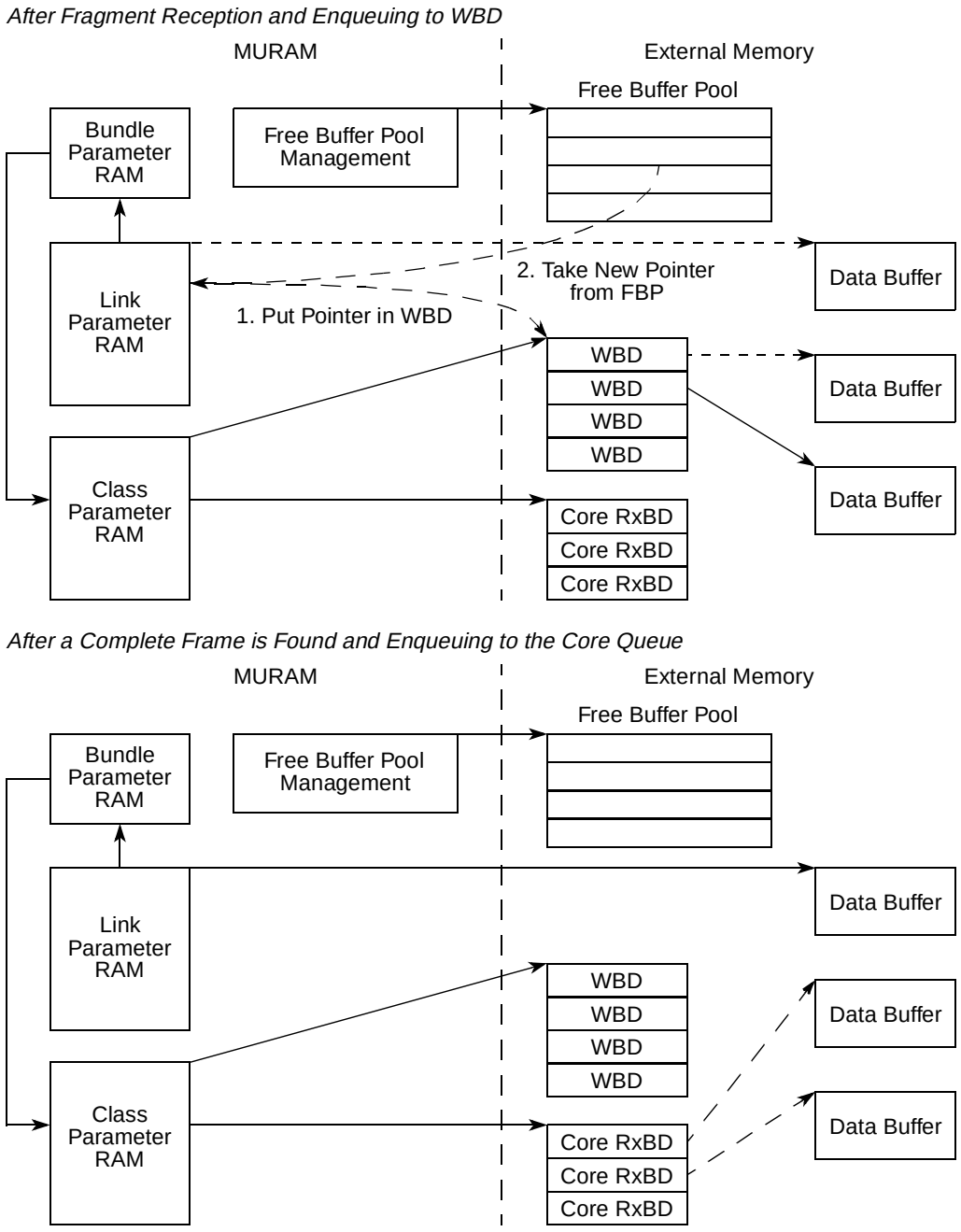


Figure 37-8. Data Buffer Pointers State After Fragment and Frame Reception

37.6.5.1 WBD and Status Ring

37.6.5.1.1 WBD Ring

The WBD structure is designated for the RISC usage and is not visible to the application. It serves as an intermediate buffer handling to store the fragments until a complete frame is assembled. Its purpose is to manage differential delays between links on a bundle so that it would be transparent to the host application.

- WBDs are located in external memory, status parameters are stored in the MURAM to simplify the management and to improve performance.
- Each WBD entry represents a single fragment.
- The status of every fragment is stored in the Status Ring.
- The allocation of a WBD entry is done according to the fragments sequence number. Before a fragment is placed in the queue, its sequence number is checked to see if it fits into the current window.
- Only complete frames will be enqueued to the Host queue or to the Demultiplexer.
- Each class has its own WBD ring.
- The size of a window must be a power of 2.

37.6.5.1.2 Status Ring

The status ring indicates which fragments were received and their status. It has the same number of entries as the WBD ring. Each entry has an associated WBD entry and is located in MURAM.

For best memory efficiency, each entry in the status ring has the size of a nibble (4 bits) and represents a fragment. The status ring is initialized to zeros (if working in FBP mode). When a fragment is received, the Valid bit is set and the bits representing begin and end of frame are updated. The fourth bit is the Full bit, indicating if there is a valid data buffer pointer in the WBD entry. This bit is set by the RISC in case of fragment invalidation.

The Class Parameter Table (CPT) holds a pointer to the beginning of the status ring (Status Ring Base). The management of the fragments reassembly process requires two pointers:

- BF_index—Begin Fragment Index which represents the beginning of current frame.
- EF_index—Empty Fragment index which represents the next empty fragment. Both pointers are stored in CPT.

The number of entries in the status ring (like the WBD ring) should be a power of 2.

Figure 37-9 illustrates the WBD ring and the status ring:

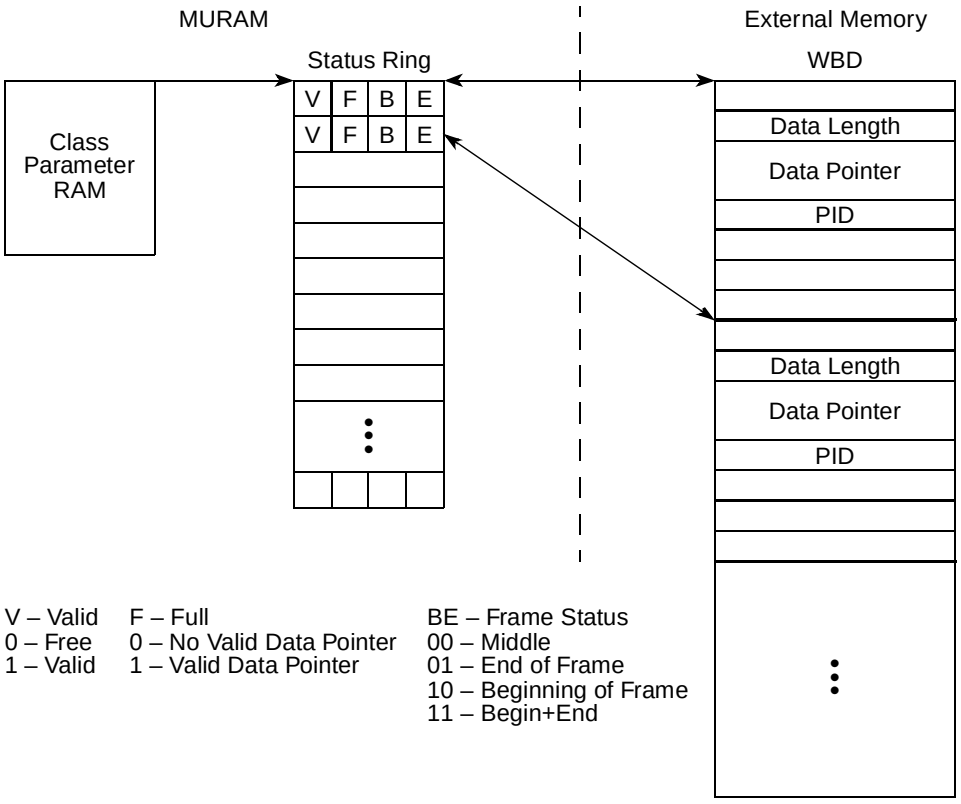


Figure 37-9. Status Ring and WBD Ring Correlation

37.6.5.2 Complete Frames Handling

Whenever a fragment whose sequence number corresponds to the location of the first empty index (EF_index) is received, the process of searching a complete frame is triggered. The process scans through the status table and updates the First_Empty_index parameter to a new value. As the scan is ongoing on the status ring, each entry is checked for its valid bit and B/E bits. When a correct and complete sequence of middle fragments and end fragment is found, a complete frame is found. The first fragment of the frame is pointed to by Begin_fragment_index. The location of Begin_fragment_index and the number of fragments are registered for the back-end process—the packets reconstruction task, via the Packets Reconstruction Table (refer to Section 37.16.2.1, “Rx Packets Reconstruction Entry”).

After a complete frame is found the Begin_fragment_index is updated to the next beginning of frame and the first empty index (EF_index) moves on till the next found end of frame. The process stops when EF_index reaches an empty fragment. The scan process can detect multiple completed packets, but it fills an entry in the Packet Reconstruction Table for each packet.

If an unexpected fragment is encountered during the process, such as B when E or middle are expected, all fragments starting from Begin_fragment_index up to the next valid Begin_fragment_index are invalidated.

The invalidation of a fragment is done by clearing the Valid bit, the Begin bit and the End bit and also by setting the Full bit in the appropriate status nibble. When a fragment is invalidated, the data pointer is not

returned to the Free Buffer Pool. On the next time a data buffer pointer is required, this pointer will be used instead of a pointer out of the FBP.

Figure 37-10 shows the process.

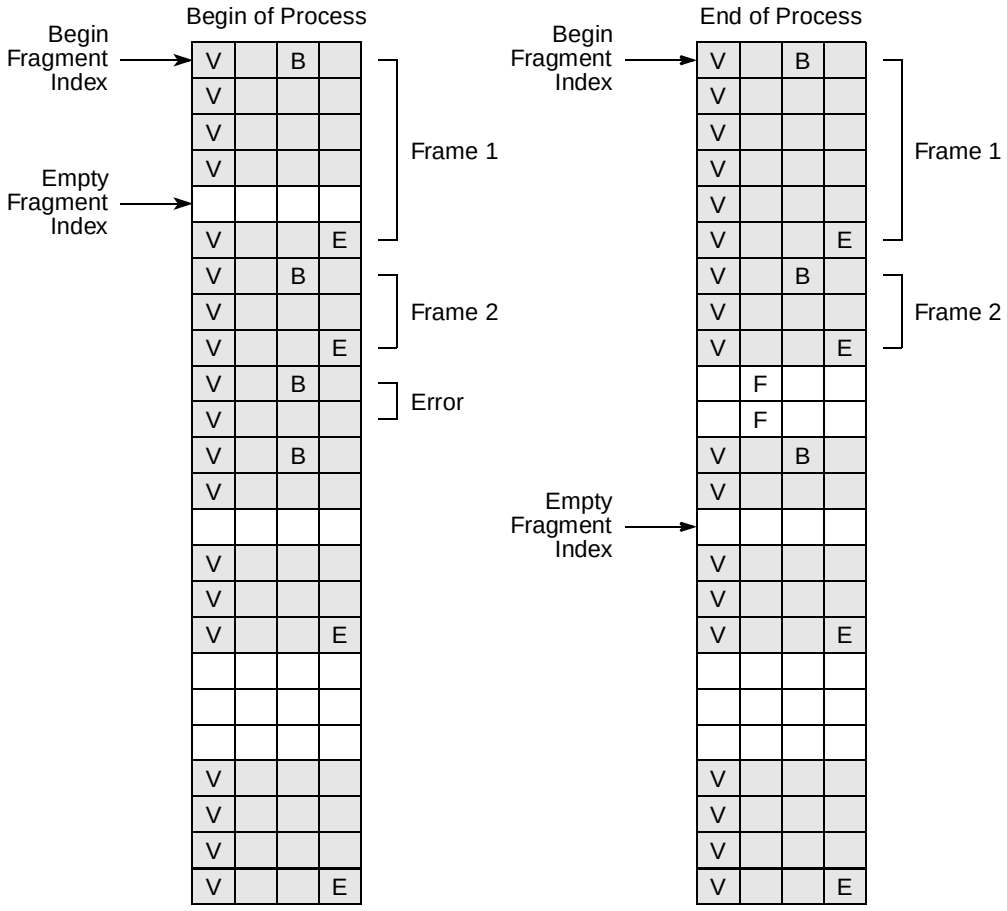


Figure 37-10. Status Ring Pointers Handling

37.6.5.3 Fragment Loss Process

RFC 1990 suggests the following algorithm: the receiver keeps track of the incoming sequence numbers on each link in a bundle and maintains the current minimum (M) of the most recently received sequence number over all the member links in the bundle. The minimum and the tracking of the incoming sequence numbers are kept per class. The receiver records the sequence number of fragments marked E. The sequence numbers are represented as offsets in the WBD ring. It always keeps the lowest sequence number of an ending fragment of a not complete frame (E). A fragment loss is detected when M advances past E.

The goal of the process is to identify fragment loss as soon as possible in order to prevent handling of many frames at a time. The receiver indicates the next empty fragment that is expected (EF_index). When M increments past the sequence number of an empty fragment, a fragment loss can already be detected. There is no need to wait for M to advance past E: since M is higher than the sequence number of the next empty fragment, and since the sequence numbers are always incrementing, the empty fragment is not expected on the link anymore. This condition also triggers a fragment loss process.

After a fragment loss is detected, Begin_fragment_index (BF_index) and Empty_Fragment_index (EF_index) are updated. First, all fragments between Begin_fragment_index and Empty_Fragment are invalidated. Empty_Fragment (EF_index) is updated to the next fragment, its status is checked. If it is empty both offsets will indicate this location. If it is B, the Begin_fragment_index is updated to it and Empty_Fragment_index increments to the next fragment location. The process continues until Empty_Fragment_index reaches an empty fragment location.

The algorithm described above works well when the fragments of the same class are distributed on all links of a bundle. If they are concentrated on fewer links, the value of M may stay unchanged because M is calculated on all the links in the bundle. This could cause the WBD ring to overflow in case of fragment loss. (Since M is the minimum of recent sequence numbers of ALL links, M will point to the last sequence number, received on the links, on which fragments are not transmitted.) To prevent that from happening, an additional mechanism is defined. A MX parameter holds the maximum value of recently received sequence number of all links. LSQN holds the last contiguous sequence number (i.e the sequence number before the one, EF_index represents) A threshold value is defined—SQN_TH. Whenever MX exceeds LSQN by more than the threshold, a fragment loss is detected. The frame containing the lost fragment (indicated by EF_index) will be discarded: BF_index will be moved to the next B, EF_index will be moved to the next empty fragment after BF_index, LSQN, M are updated.

37.6.5.4 Packet Reconstruction Task (PRT)

The packet reconstruction task (PRT) collects complete frames from the WBD ring and enqueues them to their destination queues. In the case of a muxed PPP frame, the PRT enqueues the frame to the demultiplexer queue—the demux fragment table. The PRT is triggered in case of fragment loss as well. Its purpose in that case is to clean all old fragments, collect complete frame, and synchronize the fragment pointers required to the continue of fragment reception and fragment loss monitoring.

This process is done in the background and is triggered by the front end process, utilizing a special interface which is described later. The information is passed to this task via a dedicated table—the Packets Reconstruction Table. The front end process writes a Packet Reconstruction Table entry when it wishes to trigger a complete frame search. The packet reconstruction task gets an entry from the table and searches for a complete frame. The frame is enqueued to the host or to the demultiplexer task according to the PID of the packet. If the front-end does not find a valid entry in this table, a busy interrupt is sent to the host. This could indicate some overloading in the system since the background is not able to keep up processing with the front-end.

The packet reconstruction task handles a single entry of the Packet Reconstruction Table at a time.

When enqueuing to the host queue, the task calculates how many BD’s are required for the complete frame since each fragment resides in a separate BD. Then it fetches the queue descriptor of the host. It checks if there are enough free BDs in the queue to accommodate the new frame. This check is based on the difference between the offset-in and offset-out values in the queue descriptor. If there are enough BDs, the Packet reconstruction table writes the BDs to the host queue and places the data buffers pointers into the BDs. Host processing of the BDs depends on the mode of operation. If FBP is active the host should return

the buffer pointers which were in the BDs or new buffer pointers to the Free Buffer Pool. If FBP is not active- swap mode, the host should put valid buffer pointers in the BDs which were processed. If there are not enough BDs to enqueue the complete frame, the frame is discarded and an interrupt is issued (Busy).In this case the corresponding entries in the Status Ring are invalidated (Valid bit =0); the Full bit of the entry is reset in the case of successful enqueuing and is set in case of a Busy condition.

When enqueuing to the demultiplexer, the packet reconstruction task reads an entry from the Demux Frame/Fragment Table (DFT). If this entry is valid, it enqueues a fragment to this table for de-multiplexing and continues to do so until it enqueues the last fragment of the packet. If an entry is not valid, the RISC discards all of the fragments for the reconstructed packet.

The Packets Reconstruction Table is allocated in the MURAM. The Packets Reconstruction Table size is programmable by the user, each entry representing a frame.

Figure 37-11 illustrates this process.

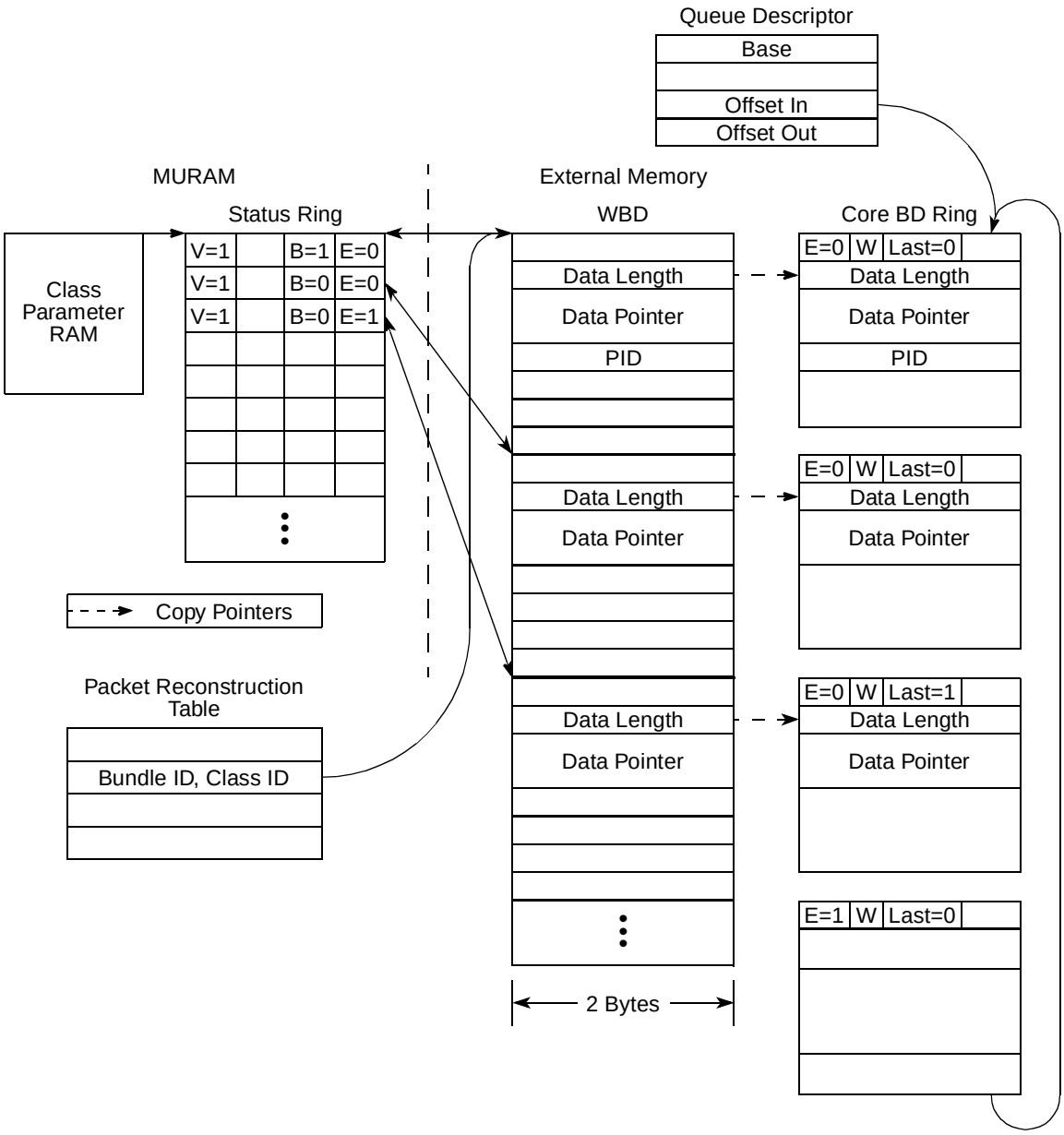


Figure 37-11. Packet Enqueuing Process

37.7 Transmission Process

37.7.1 Plain PPP Transmission

For plain PPP traffic related to a specific TDM link, utilizing Link queue (packets which are not fragmented over multilink), the framing process and the transmitting itself are done by the front-end process. The transmitter always checks if it has data to be transmitted on the Link basis; if it has none, it transmits the data for ML/MC traffic.

37.7.2 ML/MC Transmission

37.7.2.1 Fragmentation Process Structure

The process of preparing a ML-PPP fragment is also partitioned into two independent processes. The front-end deals with the actual transmission on the link, and the back-end process prepares the fragments. The background process is called the fragmentation process. Each fragment can consist of several BDs (typical when encapsulating PPP mux frames) or on the other hand may be a single BD which is transmitted over several fragments.

The back-end requires data structures to be prepared and initialized by the host. These are completely managed by the RISC during normal operation. Each bundle points to a structure of several fragments in external memory. These data structures are called Fragment Descriptors. The number of these data structures is twice the number of potential links in the bundle. Each bundle also points to a Fragment Pool Table, which manages the order of preparing and transmitting the fragments.

37.7.2.2 Fragment Descriptor Structure

Each Fragment Descriptor structure contains the information needed for transmitting a fragment. It is located in external memory and is for RISC usage only. The application should only allocate the memory required for these structures.

The user determines the size of the fragment structure according to the maximum number of BDs that constitute a fragment. For small size PPP mux subframes and a large fragment size, one can expect to have more BDs on each fragment. The size allocated must be a power of 2. The formula for calculating Fragment Descriptor size as a function of the maximum number of BDs is:

Fragment_Size = (Max No. BDs * 16) + 16.

For example, a maximum number of 7 BDs requires a 128 bytes Fragment Descriptor.

A fragment structure of 128 bytes is described in Figure 37-12 for illustration purposes only. As stated, these structures are for RISC usage only. The host should only allocate the memory for those structures.

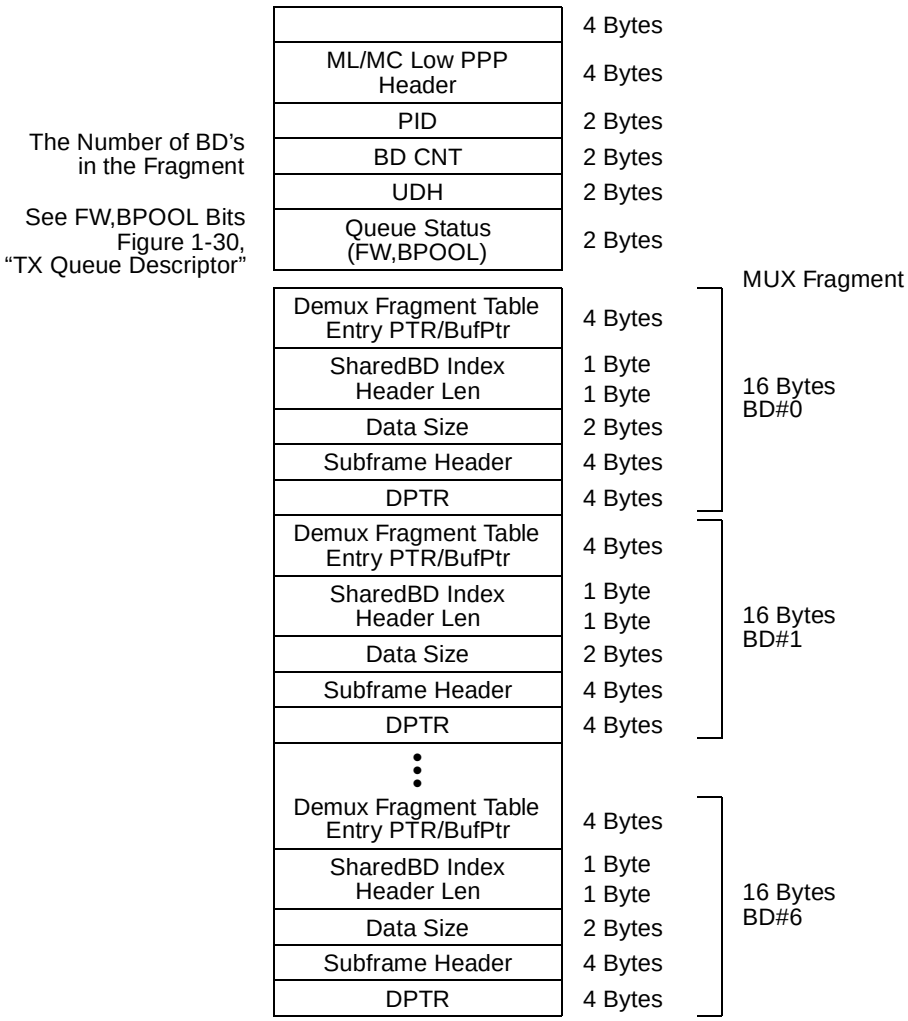


Figure 37-12. Fragment Descriptor

Figure 37-13 and Figure 37-14, illustrate the fragmentation process for two links in a bundle. Figure 37-13, part A, shows two fragments (index=0 and index=1) which are ready for transmission, therefore the Fragment Ready Count is equal to two. In Figure 37-13, part B, the first two fragments (index=0 and index=1), are transmitted by link X and link Y, and a new fragment (index=2) is ready for transmission; therefore the Fragment Ready Count is equal to one. In Figure 37-14, part C, link Y has finished to transmit fragment (index=1), and has started to transmit next fragment (Index=2), also a new fragment is ready for transmission (index=3), therefore the Fragment Ready Count is equal to one. Link Fragment Offset In is advanced, because one fragment was already transmitted, and entry 0 in the table was updated with index=1. In Figure 37-14, part D, link X has finished to transmit fragment (index=0), and started to transmit fragment (Index=3), also a new fragment is ready for transmission (index=1), therefore the Fragment Ready Count is equal to one. Link Fragment Offset In is advanced, because one fragment was already transmitted, and entry 1 in the table was updated with index=0.

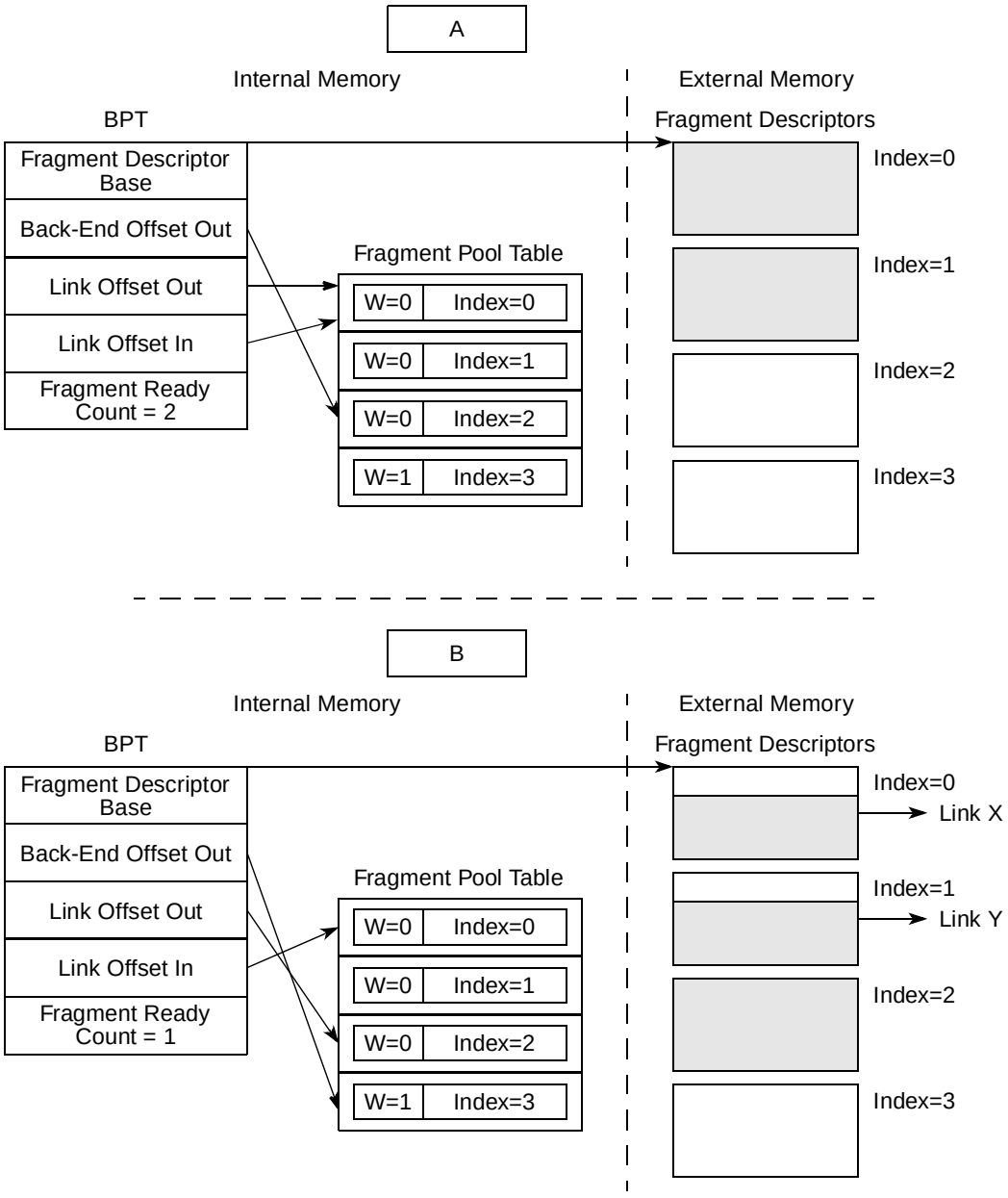


Figure 37-13. Fragmentation Structures (for 2 Links per Bundle)

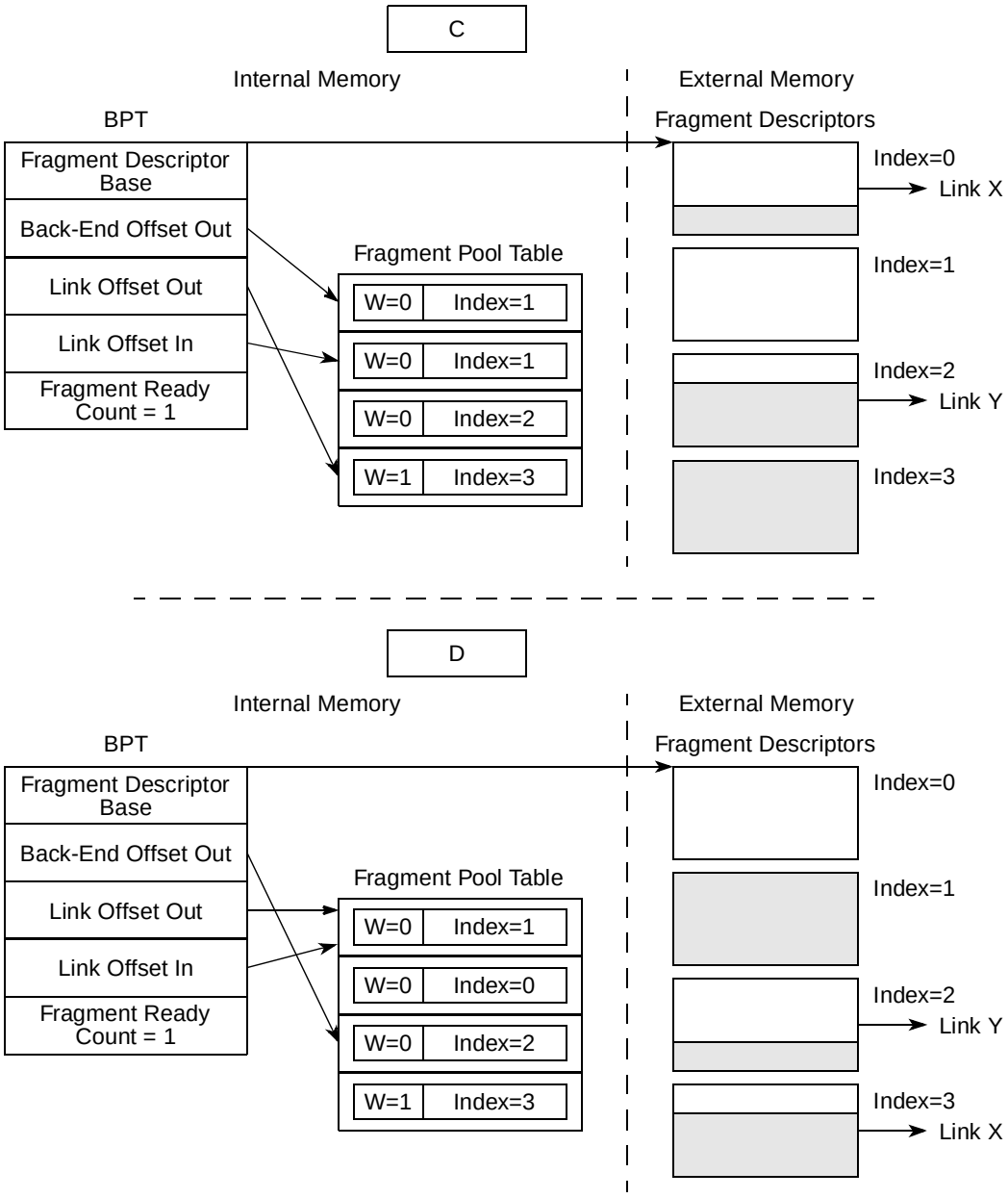


Figure 37-14. Fragmentation Structures (for 2 Links per Bundle)—Continued

37.7.2.3 Shared Buffer Handling

In ML-PPP, several links may attempt to transmit fragments from a common buffer. This happens when a packet which is larger than MRU size resides in a single buffer. This requires special handling of the BD related to this buffer. Only the last link that transmits the last fragment from this buffer should return the buffer pointer to the Free Buffer Pool.

This process is done by the RISC using a dedicated management routine. Up to 32 shared BDs can be handled by the system. The application must allocate the space for this management table for each bundle.

37.7.3 Weighted Fair Queueing (WFQ)—Selection of Queues/Classes

Weighted fair queueing (WFQ) is a mechanism for scheduling a queue or a class for transmission. The selection process is done in a few levels according to a predetermined order. As described in Figure 37-3, there are two sources for transmission, the Link and the ML/MC. Priority of the link queue is higher than ML/MC queues.

On the link queue there is an option to define up to eight queues where selection is done by the WFQ algorithm. If there is no need for multiple queues this data structure and process are omitted. On the ML/MC there is a need to select a class out of up to 8 classes. This is done also by a WFQ algorithm. The WFQ management resides in the internal ram (MURAM) for best efficiency and coherency. For each queue/class represented in the WFQ scheduling scheme there is a status bit indicating the status of the queue (i.e. is the queue full or empty). This bit is embedded in the WFQ structure. In each queue descriptor there is a pointer pointing to the location of this bit in the internal memory. In all of the cases if there is no need for multiple queues the WFQ data structures are omitted. In this case there will be only a single bit indicating queue empty/full. This bit should be handled by the host as described below. (See Section 37.7.3.5, “Management of the Queue Empty (QE) Bit”).

Each queue has an associated weight, which is translated to a parameter called INC (Increment) programmed in the WFQ data structure. INC is inversely proportional to the weight value. The higher the weight (lower the INC) value, the higher the bandwidth for the queue.

In weighted fair queueing, all queues are examined and the queue with the smallest finish time is selected. The relative data rate for the queue is determined by the WINC parameter as programmed in the Weighted Fair Queueing Table (see Section 37.5.5, “Weighted Fair Queue (WFQ) Table”). The Queue Empty bit of a queue that has no frames/fragments to transmit is set, and as a result this queue will not be selected.

The following sections describe different scheduling schemes that can be obtained by the WFQ algorithm.

37.7.3.1 WFQ Among All Queues

The WFQ mechanism can be used to implement many scheduling schemes of weighted fair queueing (WFQ) among all 8 queues, as the example in Table 37-2 shows.

Table 37-2. Example of WFQ among All Queues

	Queue0	Queue1	Queue2	Queue3	Queue4	Queue5	Queue6	Queue7
INC	3	5	2	30	2	5	30	20

The formula for the allocated bandwidth of each queue out of the total bandwidth, assuming Queue#i has an INC=Ki value associated with it, is as follows:

$$\text{QueuePriority}_i = \frac{1}{K_i} \times \sum_{\text{all queues}} \frac{1}{K_j}$$

For the above example Queue0 has priority of 18%, Queue1 receives 10%, and so on. The algorithm implements WFQ algorithm in the sense that at any given time all queues are examined, and the one with

the smallest finish time is chosen. Finish time is updated for the selected queue by adding the queue INC value to the finish time.

37.7.3.2 Strict Priorities among All Queues

Strict priority is a mode in which, whenever a high priority queue has a frame to be transmitted, this queue is chosen. To implement strict priority among all queues (Queue0 is highest and Queue7 is lowest priority) program all INC values to zero.

37.7.3.3 Round Robin Algorithm

To implement Round Robin the application must program the same INC value (greater than 0) for all queues. In this way all queues get the same relative weight.

37.7.3.4 Mixed Mode

To implement mixed mode where a number of queues are in strict priority and others are scheduled with WFQs: The strict priority queues are assigned to the low numbered queues (starting from Queue number 0—Queue0), these queues are assigned INC=0. Other queues are assigned INC according to their relative weight.

Table 37-3. Example of mixed mode WFQ

	Queue0	Queue1	Queue2	Queue3	Queue4	Queue5	Queue6	Queue7
INC	0	0	2	30	2	5	30	20

In this example Queue0 and Queue1 are strict as long as Queue0 has a queue with a BD that is ready to transmit. If it has no ready BD, Queue 1 is chosen. If it is also empty (The BD are not ready), one of Queue2 to Queue7 is chosen. according to their relative weight.

37.7.3.5 Management of the Queue Empty (QE) Bit

Each queue/class has a status indicating if it is full or empty, so that it can be scheduled for transmission if full and removed from the selection automatically, when it is empty. The RISC and the host change the QE bit according to the state of the queue. When reaching a point where there are no BD’s in the queue, the RISC sets the QE (Queue Empty or CE Class Empty) bit. The host, when filling up a queue of BD’s for transmission, should clear this bit in order to put the queue into the scheduling process. When clearing a Link/Class empty bit the host should follow the procedure described in Figure 37-15:

1. Set the BD[R] bit for the corresponding BD’s under the queue.
2. Check if the QE bit is set, and only if it is set clear it, in order to enable the queue.

The user may desire only one queue instead of 8 queues. In this case the Weighted Fair Queueing Table is not necessary. There are four possible modes that determine the location of the QE and the CE bits that are described in Section 37.5.5, “Weighted Fair Queue (WFQ) Table.”

Table 37-4. Queue Empty Bit Management

Case	PPP Mode	Number of Classes	Number of Queues	Queue/Class Empty Bit Location
1	Link	not applicable	1	LPT[LMR]->QE
2	Link	not applicable	>1	Queue WFT[QE]
3	ML	1	not applicable	BPT[BMR]->CE (Offset 0x0E in BPT)
4	MLMC	>1	not applicable	Class WFT[QE]

For example, in the case of MLMC with multiple classes (case 4 in the table), the host/RISC needs only to manage the Class WFQ[QE] bit. The QD[QE_PTR] parameter would be initialized to the relevant Class WFT entry pointer.

See the host flow in Figure 37-15.

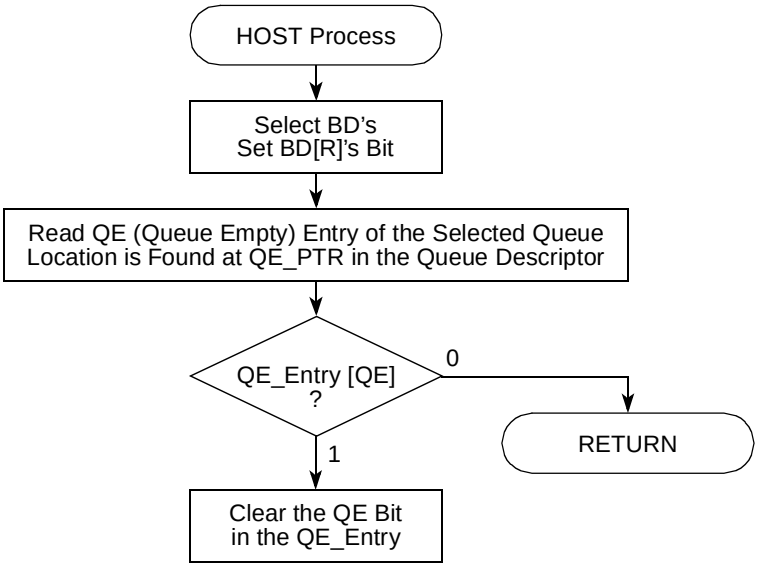


Figure 37-15. Activating a Queue by the Host

37.8 Background Processes Interface

As described earlier, both the receiver and the transmitter have background activities done in parallel to the MCC activities. Because it is possible to have a system with multiple bundles running concurrently there is a need to indicate to these background processes on which bundle their services are required.

The transmitter and receiver have different activation process because of the difference in the tasks done by each of those background processes.

37.8.1 Transmitter Background interface

The task of the transmitter background is to perform a selection between the classes in a ML/MC setup and to generate the fragments for transmission. This should be done on a bundle basis since fragments are

prepared for each bundle. For doing this there is a special table defined used by the transmitter. The table contains a list of valid bundles in the system which require the background task services. It is filled by the host. Up to four bundles are allowed in the system. Last entry is marked with a Wrap indication.

The table consists of up to four entries for four available bundles. Each entry contains a pointer to a bundle parameter table (BPT) in the system which is a ML/MC bundle. The front-end generates requests to the back-end. Upon request the back-end processes an entry in its table. It steps to the next entry upon completion of a full fragment for the current bundle. Service is done in a round robin method Figure 37-16 describes this handshake between the back-end and the front-end process.

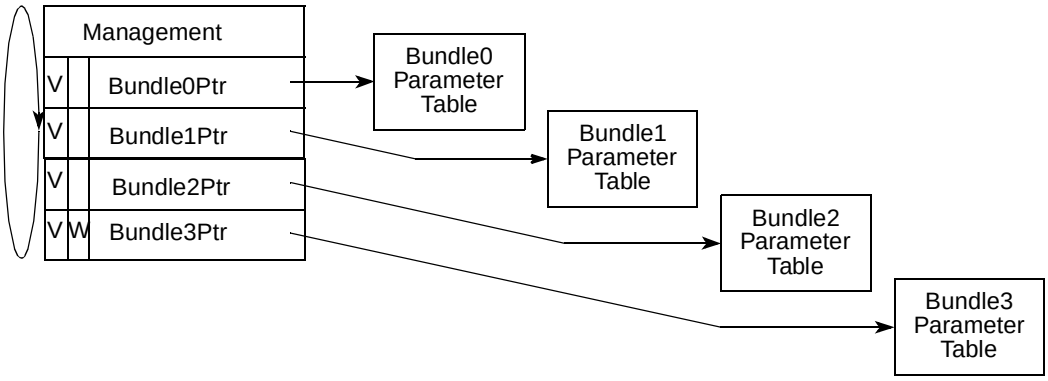


Figure 37-16. Fragmentation Process interface

37.8.2 Receiver Background interface

The packet reconstruction task which is a receiver background task provides services on a class and bundle basis. Therefore upon each packet reception which can consist of several fragments, the front end writes a request in a centralized request table (Packet Reconstruction Table). Each entry in this table contains all the information needed for the reconstruction task. The size of the table is determined by the application and last entry is marked with a Wrap indication. Management of the process is located in the MURAM.

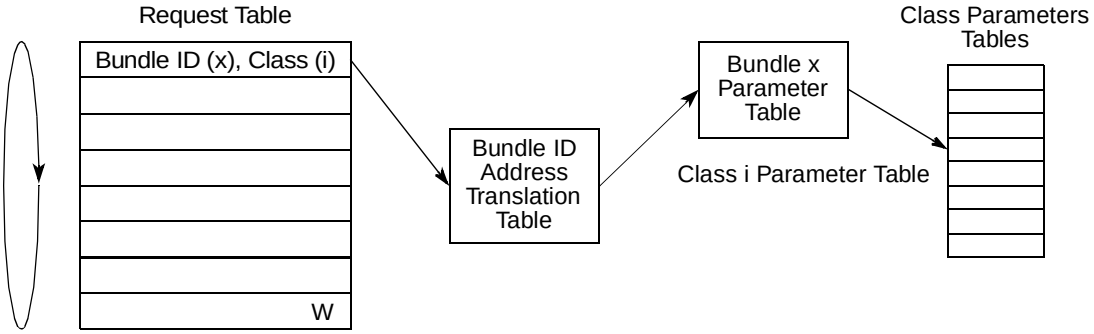


Figure 37-17. Packet Reconstruction Process interface

37.9 PPP Mux Process

The RISC is capable of processing PPP mux superframes. The process is a stand alone process having its own page. It can be looked as an upper layer of handling this service. It is called by the back-end processes of the receiver and the transmitter and has a dedicated interface and handshake for doing that.

On the ingress the RISC detects and analyzes a PPP mux superframe arriving as part of a ML/MC multi fragment superframe or on a link as a plain PPP mux packet. Once the frame is identified as a muxed frame, it is forwarded to the Demux process which parses through the frame, breaks the superframe into the encapsulated sub-frames and puts the subframes into their queues with all information needed.

On the egress side the RISC encapsulates superframes for all the queues which are listed as requiring MUX services. This process is also done in the background and upon completing a superframe it is taken by the transmitter back-end process for fragmentation into the ML/MC traffic.

37.9.1 PPP/ML-MC Demultiplexing

Three program entities are involved in the demultiplexing process—the PPP mux detection, the demux process, and the host. The receiver detects that a PPP mux frame was received based on the PID information. If the PPP mux frame was received over ML or ML/MC connections then each fragment of the frame resides in a single data buffer. All parameters of that fragment (e.g Length, Data buffer pointer, etc.) are passed to an entry in the Demux Fragment Table. A time stamp (stamped on fragment basis) is also passed to this Table (DFT). At this point the receiver requests the services of the demux process, The Demux scans through the superframe, extract the information for each of the subframes within the fragment, and assign each subframe its own BD. No data is ever being copied in this process. When the demux becomes active it first fetches a DFT entry, then a queue descriptor for a BD ring which is, in fact, empty data structures ready to be used by the RISC. The RISC processes the fragment, extracts and analyzes subframe headers and put the necessary parameters into the BDs. Once all the subframes are processed, the demux generates an interrupt to the host, containing information on the location of the queue descriptor and the DFT entry index. The host handles the BD ring. Since each BD contains a pointer which points into the original fragment buffer there is a unique way for returning a buffer to the FBP after ALL of its sub-frames BDs were processed.

Once the demux has finished processing all DFT entries it becomes inactive and will be activated by another request from the receiver.

Figure 37-18 illustrates the mechanism described above.

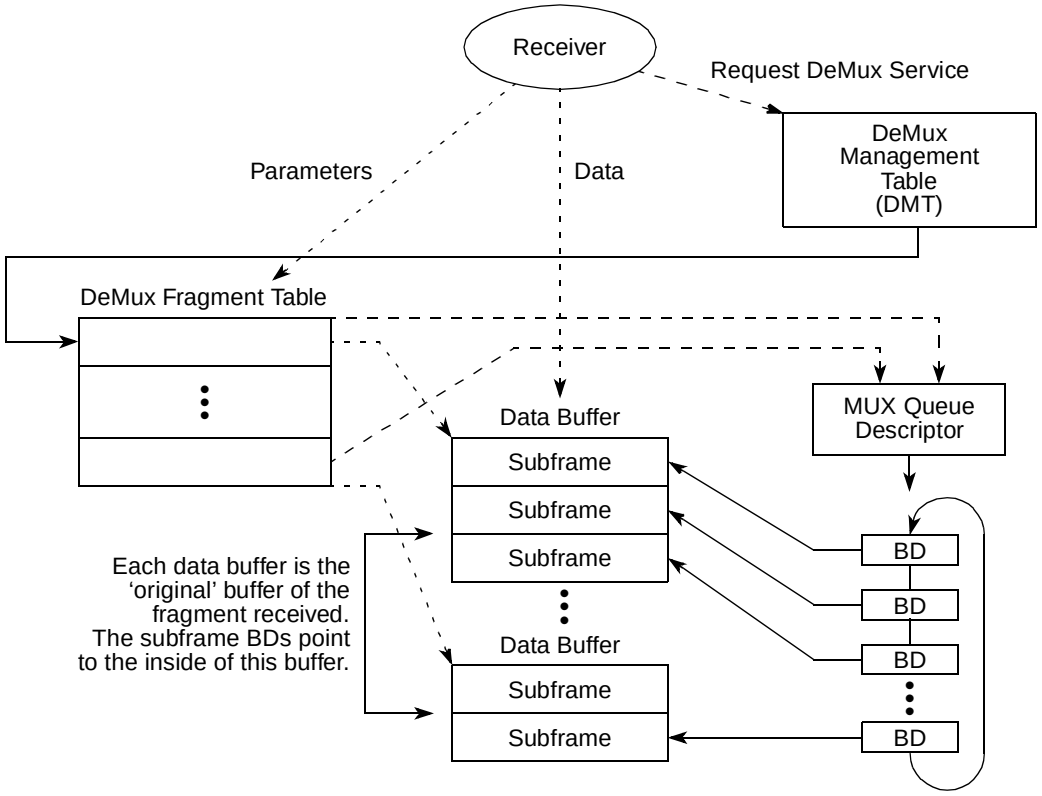


Figure 37-18. Demux Operation

37.9.2 PPP/ML-MC Muxing Encapsulation

This centralized process provides encapsulation services to all the muxed queues in the system.Note that there can be up to 16 Mux queue in the system.The host puts on each of the queues sub-frames. The mux process scans through the queues list and builds a superframe. Each of these queues can have different super-frame size and attributes. The process starts building the super-frame by scanning the BD ring of each of the queue and upon getting a valid BD it modifies the BD so it can be processed later by the transmitter back-end (or by the Link itself if this is a Link queue). This implies that there are three entities dealing with this queue—the host, the mux process, and later, the transmitter back-end or the Link itself transmitting the frame. The queue descriptor, for this reason, has a unique description. The process continues as long as the BD's are ready. When the desired super-frame size is reached or when the time-out period has expired (if enabled) the superframe is handled to the transmitter. An aging mechanism can also be activated so that an "old" packet is not encapsulated into the super-frame.

The mux process does not start generating the next superframe as long as the previous one has not been taken for fragmentation. Once this is done it has a new request and starts the process again. The list of queues for this process is located in the Mux Parameters Table which contains entries for each muxed QD. It is a static list which can be modified by the host. The mux processing scans through the list in a round robin fashion so the bigger the list is, the higher is the latency for building a super-frame. Management of processing the entries in this table is located at the Mux Management Table.

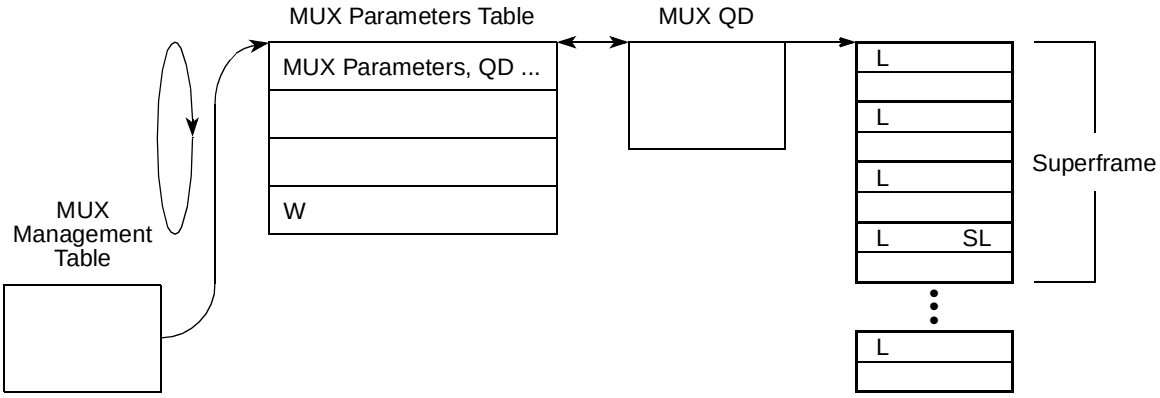


Figure 37-19. Mux Operation

37.10 Programming Model

37.10.1 Global Parameter RAM

For ML/MC operation the Global Parameter RAM is extended and has some extra parameters besides those that are already defined in the MCC page.

The following table indicates the PPP Global Parameter Ram for the MCC page.

NOTE: Important

For any pointer value to a MURAM location (noted as internal pointer) the user should put the offset to the MURAM and not the full address of the data structure. In case of an interrupt event where such a pointer is part of the information provided in the interrupt entry it is the offset value and not the address itself.

Offset + 0x90	PPP_INTBASE			
Offset + 0x94	—			
Offset + 0x98	—			
Offset + 0x9C	—			
Offset + 0xA0	Bundle ID table	Bundle0PTR		
Offset + 0xA4		Bundle1PTR		
Offset + 0xA8		Bundle2PTR		
Offset + 0xAC		Bundle3PTR		
Offset + 0xB0	Packet Reconstruction Management pointer			
Offset + 0xB4	—			
Offset + 0xB8	Tx_BG SNUM	Rx_Packet Reconstruction_SNUM	Tx_Mux SNUM	Rx_De-Mux SNUM
Offset + 0xBC	Demux Management Table base			
Offset + 0xC0	Mux Encapsulation Management Table base			
Offset + 0xC4-Offset+ 0xFF	—			

Figure 37-20. Global Parameter RAM

Table 37-5 describes the Global Parameter RAM fields.

Table 37-5. Global Parameter RAM

Offset	Name	Width	Description
0x90	PPP_INTBASE	Word (internal)	Pointer to the base address of the interrupt queue parameter tables. Should be 0x20 bytes aligned. Up to 14 queues are available.
0x94–0x9F	Reserved	12 Bytes	Should be cleared.
0xA0	Bundle0PTR	4 words	Bundle ID Table. Should be initialized only if Rx or Tx ML MC PPP operation is needed. Each entry contains a BPT offset, and the entry location is determined according to the BPT's BMR[BundleID]. e.g. entry that points to BPT with it's BMR[BundleID]=2 should be located in third place. Entry description: bit 0 -Valid bit. Should be marked by the user. An entry which has its valid bit cleared will not get serviced. bit 1 - Wrap bit. Last entry should have its Wrap bit set to 1. bits 2-7 - Reserved set to 0. bits 8-31 - BPT offset.
0xA4	Bundle1PTR		
0xA8	Bundle2PTR		
0xAC	Bundle3PTR		
0xB0	Packet Reconstruction Management pointer	Word (Internal)	Pointer to the Packet Reconstruction Management table. Valid only if ML MC PPP frames are in use or when PPP Mux frames are received. Note that even if the PPP Mux is received on the Link with out ML MC functionality, this table should be allocated. The user must assign value to the Rx_Packet Reconstruction_SNUM in these cases. Should be 64 bytes aligned and it's size is 64 bytes. Refer to Table 37-24.

Table 37-5. Global Parameter RAM (continued)

Offset	Name	Width	Description
0xB4	Reserved	Word	RISC internal use. Should be cleared.
0xB8	Tx_BG SNUM	Byte	Tx background process SNUM. The SNUM is taken from the available SNUMs described in Table 20-15. Should be set only if Tx ML-MC functionality is needed, otherwise should be cleared.
0xB9	Rx_Packet Reconstruction_SNUM	Byte	Rx Packet reconstruction process SNUM. The SNUM is taken from the available SNUMs described in Table 20-15. Should be set only if Rx ML-MC functionality is needed or if PPP Mux is enabled, otherwise should be cleared.
0xBA	Tx_Mux SNUM	Byte	Tx Mux encapsulation process SNUM. The SNUM is taken from the available SNUMs described in Table 20-15. Should be set only if Tx Mux functionality is needed, otherwise should be cleared.
0xBB	Rx_De-Mux SNUM	Byte	Rx De-Mux process SNUM. The SNUM is taken from the available SNUMs described in Table 20-15. Should be set only if Rx Mux functionality is needed, otherwise should be cleared.
0xBC	Demux Management Table base	Word (internal)	This field holds the pointer to the DeMux Management Table (DMT). The DMT resides in internal memory. User should allocate 64 bytes for this table, and it should be 64 bytes aligned. See Table 37-30.
0xC0	Mux Encapsulation Management Table base	Word (internal)	This field holds the pointer to the Mux Management Table (MMT). The MMT resides in internal memory. User should allocate 56 bytes for this table, and it should be 64 bytes aligned. See Table 37-33.
0xC4–0xFF	Reserved	-	Should be cleared.

37.11 Link Parameter Table (LPT)

Link-related parameters are stored in the Link Parameter Table (LPT). It contains a pointer to the bundle that the LPT related link is a member of, pointers, and parameters of the Link Host Queue.

The Link Parameter Table (LPT) is an extension to the MCC Channel-Specific HDLC Parameters. It is accessed on each MCC Transmit or Receive request. The size of the LPT is 256 bytes so it consumes the space of four MCC channel specific parameters. Location in the MURAM is determined by 64*CH_NUM (relative to the MURAM base address). The CH_NUM is the channel number, and must be a multiply of 4. (8 bits alignment) Any PPP host command should use this channel number as a command parameter (MCN).

NOTE

If a link is configured as a single link (not Multilink-Multiclass), the user still has to initialize the first 28 bytes (0x1C) of a Bundle Parameter Table related to this link.

Table 37-6 describes the Link Parameter Table.

Table 37-6. Link Parameter Table (LPT)

Offset[1]	Name	Width	Description
0x00	TSTATE	Word	Tx internal state. It is initialized by the host command. Refer to Table 37-44.
0x04	ZISTATE	Word	Zero-insertion machine state. (User-initialized to 0x1000_0207 for regular channel, and 0x3000_0207 for inverted channel)
0x08	ZIDATA0	Word	Zero-insertion high word data buffer (User-initialized to 0xFFFF_FFFF)
0x0c	ZIDATA1	Word	Zero-insertion low word data buffer (User-initialized to 0xFFFF_FFFF)
0x10	Bundle_Shared_ptr	Word (internal)	Bundle & Link Shared Parameters Pointer. The Bundle & Link Shared Parameters pointer points to the bundle parameter table in case of ML/MC PPP. In case of Plain PPP it points to a table of parameters that are used by the Link and by the ML so that this is an extension of the LPT table. **Note:** Should be 64 bytes aligned.
0x14	Reserved	Hword	Reserved, should be cleared.
0x16	LINTMSK	Hword	Link interrupt mask flag. See Figure 37-21.
0x18	LMR	Word	Link Mode Register. See Table 37-7.
0x1C	Reserved	Word	Reserved, should be cleared.
0x20	RSTATE	Word	Rx internal state. It is initialized by the host command. Refer to Table 37-44.
0x24	ZDSTATE	Word	Zero-deletion machine state (User-initialized to 0x80ffffe0 for regular channel and 0xA0FF_FFE0 for inverted channel)
0x28	ZDDATA0	Word	Zero-deletion high word data buffer (User-initialized to 0xFFFF_FFFF)
0x2C	ZDDATA1	Word	Zero-deletion low word data buffer (User-initialized to 0x8000_FFFF)
0x30	Reserved	Word	Reserved, should be cleared.
0x34–0x3B	Reserved	12 Bytes	Reserved, should be cleared.
0x3C	Rx_Swp_Data_ptr	Word	If a FBP is used by the receiver this entry should be initialized with the value 0xFFFF_FFFF. In case when only some of the RxQDs under this link use FBP mode and other RxQDs uses swap mode, then the host still need to initialize this entry with the value 0xFFFF_FFFF, and the buffers pointers allocated for the swapped RxQDs should match the size of the buffer pointers in the FBP. Once the INIT PPP Rx link (see 0xPPP Commands) command is issued this value is changed by the RISC to a valid data pointer from the FBP. It must be used as an acknowledge for enabling traffic on a link If a FBP is not in use by the receiver, this pointer is swapped with the existing pointer in the BD. It should be initialized with a valid data pointer. In this case the WBD and status ring should be initialized as described in 37.16
0x40	MRU	Hword	Maximum Receive Unit bytes length. Must be bigger than 16 bytes. For Plain PPP it includes the information field and any padding; but not the protocol, address, control or FCS fields.
0x42	Rx_ACC	Hword	Receive Address and Control Characters.

Table 37-6. Link Parameter Table (LPT) (continued)

Offset[1]	Name	Width	Description
0x44	Tx_LWFQ_ptr	Word (internal)	Transmit Link Weighted Fair Queue Pointer. This entry points to the Link Weighted Fair queue structure, which occupies 32 bytes, and should be 32 bytes aligned. This parameter should be initialized to 0 if there are not any queues for this link or there is a single queue for this link (in this case the link is located in Tx_LQDB parameter).
0x48	Tx_LQDB	Word (external or internal)	Transmit Link Queue Descriptor Base Pointer. Point to the base address of the Link queue descriptors. Could be address to external or internal memory. Should be 16 bytes aligned. When working in WFQ mode the Link queue descriptors are placed consequently and each queue occupies 16 bytes. When working in a single queue mode, if LMR[LQE]=1 the Link queue is empty and therefore will not be scheduled for transmission. When the user clears this bit the queue will be scheduled with highest priority.
0x4C	Reserved	Word	Reserved, should be cleared.
0x50	Rx_IB_ptr	Word (internal)	Points to the Receive Internal Buffer. The size and alignment of this buffer is determined by the Rx_IB_Size entry in this table. The buffer enables parsing of received data before copying to external memory.
0x54	Tx_LInt_Frg_Ptr	Word (internal)	Transmit Link Internal Fragment Base Pointer. Allocate 64 bytes in Multi-user RAM for the current fragment which is transmitted by the link. Should be aligned to 64 bytes.
0x58-0x73	Reserved	32 Bytes	Reserved, should be cleared.
0x74	Rx_LCP_Data_ptr	Word	Rx data pointer used by RISC to receive LCP packets. When the receiver identifies this type of packet it will use this pointer to store the data. The LCP data buffers may differ in size from the data buffers, therefore after receiving a complete frame, this pointer is swapped with a buffer pointer from the LCP Queue. Note: The data buffer, which the Rx_LCP_Data_ptr refers to, should fit the LCP queue data buffer size.
0x78	Rx_LMuxQDP	Word (external or internal)	Rx Link Mux Queue Descriptor pointer. Can reside in external memory or internal memory. Points to the Queue Descriptor of the Link Mux Queue. When a common queue descriptor for all the links in the bundle is used, the Queue Descriptor should reside in internal memory. All plain PPP mux messages will be stored in this queue by the Demultiplexer.
0x7C	Int_TxQD_PTR /Cur_IntQD_PTR	Word (internal)	Internal Tx QD Pointer. The user should allocate 16 bytes for this entry. Pointer to Multi-user RAM for the location where the external Tx QD is fetched, when external Tx QD is used (LMR[TxExtQ]=1). Should be 16 bytes aligned. If LMR[TxExtQ]=0 the user should initialize to 0.
0x80	Rx_LQDP	Word (external or internal)	Rx Link Queue Descriptor pointer- Points to the receiver Queue Descriptor of the Link. Can reside in external memory or internal memory. When a common queue descriptor for all the links in the bundle is used, the Queue Descriptor should reside in internal memory. All plain PPP messages will be stored in this queue.
0x84	Reserved	Word	Reserved, should be cleared.

Table 37-6. Link Parameter Table (LPT) (continued)

Offset[1]	Name	Width	Description
0x88	Rx_LSR	HWord	RISC usage. Internal state of the RISC when processing a PPP packet. Initialize to 0x0007
0x8A	Rx_IB_Size	Byte	Receive Internal Buffer Size. Initialized by the user to 32.
0x8B	Reserved	Byte	Reserved, should be cleared.
0x8C–0x9B	Reserved	16 Bytes	Reserved, should be cleared.
0x9C	Link_Statistics_Table_Ptr	Word (internal)	Pointer to a table in the Multi-Port RAM which contains statistic information about this link. See Table 37-8 for details. The address should be 0x40 byte aligned.
0xA0–0xAB	Reserved	12 Bytes	Reserved, should be cleared.
0xAC	Rx_LState	Word	RISC usage. Internal state of the RISC when processing a PPP packet. It is initialized by the host command. Refer to Table 37-44
0xB0	Rx_CState	Word	RISC usage. Internal state of the RISC when processing a PPP packet. It is initialized by the host command. Refer to Table 37-44
0xB4	Reserved	Word	Reserved, should be cleared.
0xB8	Rx_LQDP_internal	Word (internal)	Internal Rx QD Pointer. The user should initialized this field in case of LMR[RxExtQ]=1 or in case of Mux traffic on the link. The user should allocate 32 bytes. Should be 32 bytes aligned.
0xBC-0xFF	reserved	68 Bytes	Reserved should be cleared.

[1] The offset is relative to Multi-user RAM base address + 64*CH_MUM

37.11.1 Link INTMSK Register (LINTMSK)

The link interrupt mask register (LINTMSK) is a user-initialized register, shown in Figure 37-21.. It provides enabling/disabling events defined in the Interrupt queue table entry.

	0	1	2	3	4	5	6	9	10	11	15
Interrupt entry	—		MRU	—	NIDL	IDL	—		UN	—	
Mask	—		Mask bit	—	Mask bit	Mask bit	—		Mask bit	—	

Figure 37-21. Link INTMSK Register (LINTMSK)

To enable an interrupt, set the corresponding bit in this register. If a bit is cleared no interrupt will be issued.

37.11.2 Link Mode Register (LMR)

The link mode register (LMR) is a user-initialized register, shown in Figure 37-22.

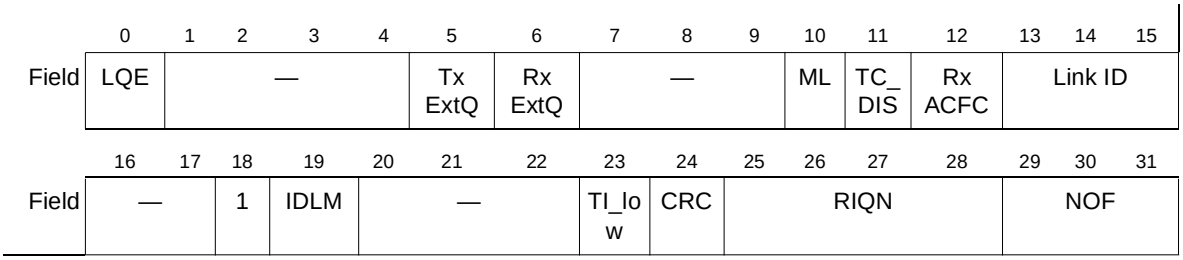


Figure 37-22. Link Mode Register (LMR)

LMR fields are described in Table 37-7

Table 37-7. LMR Field Descriptions

Bits	Name	Description
0	LQE	Link Queue empty bit. Initialized to 1. Valid only when there is one queue per Link in this bundle. 0 The Link queue is not Empty. Cleared by the core only after setting the [R] bit. 1 The Link queue is empty. Set by the RISC when there is not Valid buffer in the queue. **Note:** If the Link queue is a MUX queue, this bit will be handheld automatically by the RISC. The host shouldn't change this bit after initialization to 1. The MUX encapsulation process will clear this bit upon completion preparation of a super-frame.
1–4	—	Reserved.
5	TxExtQ	Transmit external link queues. 0 The Link queues are located in internal memory. 1 The Link queues are located in external memory.
6	RxExtQ	Receive external link queues. 0 The Link queues are located in internal memory. 1 The Link queues are located in external memory.
7-9	—	Reserved. Should be cleared.
10	ML	Tx MultiLink mode. It defines if there is a background process for fragments generation. 0 There is no ML/MC traffic generation e.g. there is no need for fragmentation process. Fragments will not be generated when this bit is cleared. 1 There is ML/MC traffic generation on the Tx ML bundle pointed by the LPT(BNDLPTR) parameter.
11	TC_DIS	Tx CRC disabled. 0 Transmit the CRC sequence after the last data byte. 1 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.
12	RxACFC	Receive ACFC (address and Control Field Compressed) mode. 0 ACFC disabled on receive side. 1 ACFC enabled on receive side.
13–15	LINK ID	Link ID. The host should assign each link with a serial number. This number will be used when a link is part of a bundle. **Important**: The number of links supported per bundle is limited to 8 or 16. When assigning a link ID to a link which is a part of a bundle the application should be aware that only the 3 or 4 least significant bits are taken for fragment lose detection as appears in Table 37-15. This implies having on each bundle up to 8 or 16 links with non overlap in the 3 or 4 lsb.

Table 37-7. LMR Field Descriptions (continued)

Bits	Name	Description
16–17	—	Reserved. Should be cleared.
18	1	Must be set.
19	IDLM	Idle Mode. 0 No idle patterns are sent between frames. 1 At least one idle pattern is sent between adjacent frames.
20–22	—	Reserved. Should be cleared
23	TI_low	Transmit Interrupt low event register. Use lower part of MCCE. 0- The interrupts associated with a Link events resides in queue #4 1- The interrupts associated with a Link events resides in queue #11 (0xB)
24	CRC	Selects type of CRC. 0 16 bit CCITT-CRC 1 32 bit CCITT-CRC
25–28	RIQN	Receiver interrupt queue number. Specifies the interrupt queue number. On the link level events are logged into the interrupt queues associated with the link. It is not related to the bundle. Numbering of the interrupt queues is described in Section 37.20.4, “MCC Event and Mask Registers” which is different than that of the legacy MCC. Since interrupt queues 4,5 and 11,12 are designated for transmitter usage, the interrupt queues that are available for the receiver are 0-3, 6-10 and 13: 0000 Rx Interrupt queue number 0 0001 Rx Interrupt queue number 1 0010 Rx Interrupt queue number 2 0011 Rx Interrupt queue number 3 0110 Rx Interrupt queue number 6 0111 Rx Interrupt queue number 7. 1000 Rx Interrupt queue number 8 1001 Rx Interrupt queue number 9. 1010 Rx Interrupt queue number 10 (0xA) 1101 Rx Interrupt queue number 13 (0xD)
29–31	NOF	Number of flags. NOF defines the minimum number of flags before frames 000 At least 1 flag 001 At least 2 flags ... 111 At least 8 flags.

37.11.3 Link Statistics Table

This table contains statistics information for the specific link in the multi link system.

Table 37-8. Link Statistics Table

Offset[1]	Name	Width	Description
0x00	Rx_FCSECNT	Word	FCS error counter. Initialized to 0.
0x04	Rx_MRUCNT	Word	Longer than MRU frame/fragment counter. Initialized to 0.
0x08	Rx_LQBSY_CNT	Word	Link Queue Busy counter: Discarded frames counter due to lack of empty BDs in the Host BD Ring. Initialized to 0.

Table 37-8. Link Statistics Table (continued)

Offset[1]	Name	Width	Description
0x0C	Rx_FRM_CNT	Word	PPP plain Frames counter. Initialized to 0.
0x10	Rx_FRG_CNT	Word	Multi Link Fragments counter. Initialized to 0.
0x14	Rx_BYTCNT	Word	Byte Count - counts the number of bytes received on link. Initialized to 0.
0x18	Tx_FRM_CNT	Word	Transmitter Frame counter. Initialized to 0. Counts the number of transmitted frames on this link.
0x1C	Tx_FRG_CNT	Word	Transmitter Fragment counter. Initialized to 0. Counts the number of transmitted fragments on this link.
0x20	Tx_Byte_CNT	Word	Transmitter bytes counter. Initialized to 0. Counts the number of transmitted valid frame/fragment bytes from this link.
0x24	Rx_AB_FCNT	Word	Counts the number of frames received with abort occurrence. Initialized to 0.
0x28	Rx_ILL_FCNT	Word	Counts the number of illegal frames. Illegal frame example include: short fragment with less than 13 bytes. Initialized to 0.
0x2C	Rx_IACCNT	Word	Invalid Address & Control character counter. Initialized to 0.

1 Offset from base address pointer

37.12 Bundle Parameter Table (BPT)

The Bundle Parameter Table contains bundle related parameters and a pointer to the Class Parameter Table. User should allocate 0x80 bytes for the BPT in case of ML MC mode of operation.

The first 28 bytes of the Bundle Parameter Table (up to offset 0x1C) is common to the bundle and to the links which are members of this bundle. This bytes should be initialized even when ML MC mode is not enabled, since they are used as extension to the LPT.

Table 37-9 describes the RX Bundle Descriptor fields.

Table 37-9. Bundle Parameter Table Fields

Offset	Name	Width	Description
0x00	Rx_LCP_QD_ptr	Word (Internal)	Pointer to the queue descriptor for the LCP packets for all the links in the bundle. This QD must reside in internal memory.
0x04	Rx_LCP_MRU	Hword	Maximum size of an LCP frame on a link. The buffers on an LCP QD should be of this size.
0x06	—	Hword	Reserved. Should be cleared.
0x08	TXFBPT_BASE	Word (Internal)	Points to the TX Free Buffer Pool Parameter Tables Base. Up to 4 different FBP can be supported for the transmitter, and the actual address of each table is calculated by: FBP parameter table = TXFBPT_BASE + TQD[Tx_FBP_Sel]. Each FBP management table occupies 16 bytes.
0x0C	BMR	word	Bundle Mode Register. See Section 37.12.1, “Bundle Mode Register (BMR).”

Table 37-9. Bundle Parameter Table Fields (continued)

Offset	Name	Width	Description
0x10	TxACC	Hword	Transmitter address and control characters. According to RFC1662 should be initialized to 0xFF03. Only 2 bytes length of Address and Control characters are supported.
0x12	De-Mux Default PID	Hword	User should initialize the De-Mux Default PID. This is the default PID for usage of the De-mux process when PFF=0 in a sub-frame header.
0x14	Tx MUX UDHBase	Word	MUX User Defined Header Base, resides in external memory. Address to the UDH is (UDH_Index*0x100+Bundle_UDH_Base). The Index parameter is taken from the Queue Mode field in the Queue Descriptor. See Figure 37-32.
0x18	Rx_FBPT_BASE	Word (Internal)	Points to the Rx Free Buffer Pool Parameter Table Base. This is the FBP which is used for ML/MC traffic. Should be 0x10 bytes aligned. Each FBP management table occupies 16 bytes. Buffer's size allocated should be big enough to accommodate the maximum size specified in LPT[MRU].
0x1C	CPT_BASE	Word (Internal)	Class Parameter Table Base. Pointer to the first Class Parameter Table of this bundle. Points to the base address of Class 0 Parameter Table. Should be 128 bytes aligned. When WFQ mode is used to select class, the CPT should be located consequently, each one occupies 0x80 bytes.
0x20	Rx_CLS Lookup Table PTR	Word (Internal)	Pointer to a table that indicates which classes are available in this bundle. Since only 8 classes are supported, in case of 4 bits class number, this table also converts the 4 bits class to 3 bits class. This is a 4 entries table in case of 2 bits of class and this is a 16 entries table in case of 4 bits of class. Each entry in both cases is a single byte. see Section 37.12.2, "Class Lookup Table."
0x24	Tx_CWFQ_ptr	Word (Internal)	Transmit class weighted fair queue pointer. This entry points to the Class Weighted Fair queue structure, and should be 32 bytes aligned. In case there is a single class in this bundle this parameter should be initialized to 0. In this case the class is located in **CPT_BASE** parameter. Also the CMR[CLSID] of the class must be equal to zero.
0x28	—	Hword	Reserved. Should be cleared.
0x2A	—	Byte	Reserved. Should be cleared.
0x2B	Tx_FD_Size	Byte	Fragment Descriptor Size. Initialized to the power of two of the fragment descriptor size (for example if initialized to 7 then the actual size is 128). This size is determined by number of optional BD's in one fragment. The formula of calculating fragment descriptor size as a function of maximum number of BD's is as follows: Fragment_Size = (Max No. BD's * 16) + 16. Note: the maximum value of this parameter is 0x0B.
0x2C	No_Links	Byte	Number of links. User initialized to the number of links, which are part of the bundle.
0x2D	—	Byte	Reserved. Should be cleared.
0x2E–0x4F	—	34 bytes	Reserved. Should be cleared.

Table 37-9. Bundle Parameter Table Fields (continued)

Offset	Name	Width	Description
0x50	Tx_FDB	Word	User initialized to Fragment Descriptor Base. Pointer to the first Fragment Descriptor, which is located in the external memory. The Maximum number of the Fragment Descriptors is the number of links belonging to this bundle multiplied by 2. Should be 8 bytes aligned.
0x54	—	Word	Reserved. Should be cleared.
0x58	Tx_ShB_mngt_tbl_Base	Word (Internal)	Shared Buffer management Table Pointer. Pointer to an array of 32 Counters, each of a single byte. This is for the internal usage of the RISC and controls the shared Buffer mechanism of returning the buffer pointer back to the Free Buffer Pool. Should be 32 byte aligned.
0x5C	—	Hword	Reserved. Should be cleared.
0x5E	MFS	Hword	Max fragment size that is transmitted over the link. This includes the information field and any padding; but not the protocol (ML PID), address, control or FCS fields.
0x60	MRRU	Hword	Maximum Reconstructed received frame.
0x62–0x6F	—	14 bytes	Reserved. Should be cleared.
0x70–0x7F	Fragment Pool Table	16 bytes	Fragment Pool Table. Allocate 2*number of Links bytes (for a maximum of 8 links in a bundle, we get 16 bytes). Contains information for managing the order of preparing and transmitting the fragments. The Fragment Pool Table should be initialized with sequential index numbering of the Fragments. see description in Section 37.12.3, “Fragment Pool Table.”

37.12.1 Bundle Mode Register (BMR)

The Bundle Mode Register (BMR) is a user-initialized register, shown in Figure 37-23.

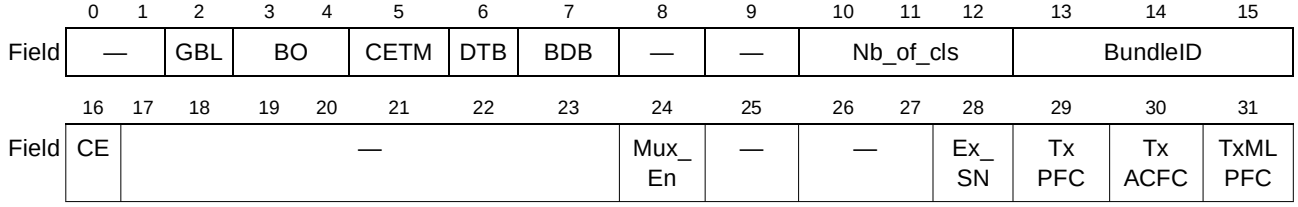


Figure 37-23. Bundle Mode Register (BMR)

BMR fields are described in Table 37-10.

Table 37-10. BMR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	GBL	Global. Setting GBL enables snooping of data buffers, BD’s, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, “Serial DMA Mode Register (SDMR).”

Table 37-10. BMR 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 or at the beginning of the next BD. 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 Figure 19-1 and Section 5.4.3.7, “Debug Configuration.”
6	DTB	Data buffers bus Selection 0 Resides on the coherent system bus. 1 Resides on the secondary bus.
7	BDB	WBD, Interrupt, Queue Descriptor, Free Buffer Pool bus Selection 0 Resides on the coherent system bus. 1 Resides on the secondary bus.
8	—	Reserved. Should be cleared.
9	—	Reserved. Should be cleared.
10–12	Nb_of_cls	Number of Classes associated to this bundle.
13–15	BundleID	Bundle ID. Distinguishes between several bundles. It is applicable for interrupt information, and for the management of the TxBundleIDTable (see Table 37-5).
16	CE	Class empty bit. Initialized to 1. Valid only when there is a single Class under the Bundle. 0 - The Class is not Empty. Cleared by the core only after setting the BD[R] bit within a queue under this class. 1 - The Class is empty. Set by the RISC when there is no data ready in the queue under the class. **Note:** If the class queue is a MUX queue, this bit will be handheld automatically by the RISC. The host shouldn’t change this bit after initialization to 1. The MUX encapsulation process will clear this bit upon completion preparation of a super-frame.
17–23	—	Reserved. Should be cleared.
24	MUX_En	Defines if there is a Tx PPP MUX queue operation. 0 No Tx PPP MUX process needed. 1 Tx PPP MUX process is needed and should called by the Link front-end process.
25	—	Reserved. Should be cleared.
26–27	—	Reserved. Should be cleared. Used by the RISC
28	Ex_SN	Extended Sequence Number Mode 0 12 bits sequence number and 2 bits for CLS id 1 24 bits sequence number and 4 bits for CLS id
29	TxPFC	Transmit PID Compressed for the Original PPP packet. (inner PID of the fragment) 0 PID transmitted is 2 bytes length. 1 PID transmitted is 1 byte length (only if MSB byte of the PID is 0).

Table 37-10. BMR Field Descriptions (continued)

Bits	Name	Description
30	TxACFC	Transmit ACFC (address and Control Field Compressed) mode. 0 ACFC disabled on transmit side. 1 ACFC enabled on transmit side.
31	TxMLPFC	Transmit ML header PID Compressed. 0 PID is 0x003d. 1 PID is 0x3d.

37.12.2 Class Lookup Table

The Class Lookup Table, shown in Figure 37-24, indicates which classes are available in this bundle in order to filter an unwanted incoming traffic. Fragments with a non-valid class number will be discarded. Also this table maps the class number in ML/MC-PPP header to the actual class number that is implemented. This mapping is required since 4 CLS bits allow 16 classes but the receiver supports only up to 8 classes. The table is 4 bytes long - single byte per entry in case of 2 bits of CLS in ML/MC PPP header or it is 16 bytes long - single byte per entry in case of 4 bits of CLS in ML/MC PPP header. The access to the table is according to the CLS of the ML/MC PPP header.

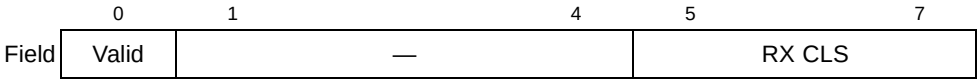


Figure 37-24. Class Lookup Table Entry

Table 37-11. Class Lookup Table Entry Fields

Bits	Name	Description
0	Valid	0 - This class is not valid in this bundle. 1 - This class is valid in this bundle.
1–4	—	Reserved. Should be cleared.
5–7	Rx CLS	The class number which is used by Rx.

37.12.3 Fragment Pool Table

This table is used by the transmitter front-end and back-end processes. The size of this data structure in bytes is twice the number of links in the bundle. Each entry contains an index to the fragment descriptor currently being used by each of the processes. An example is described Figure 37-14.

The user should allocate enough entries to accommodate the maximum number of links in the bundle. The table should be initialized in a sequential order. Last entry should be marked with a Wrap bit. See Table 37-12 for a 2-link bundle. In cases where the user would like to dynamically add links to a bundle, allocate and initialize a table which fits the maximum links possible in this bundle.

Table 37-12. Fragment Pool Table

0x00
0x01
0x02
0x83

NOTE

Since each entry is associated with a fragment descriptor which reside in external memory, the higher the number of entries (above the minimum required= 2* number of links) in the table the bigger is the memory allocated externally for the fragment descriptors.

37.13 Class Parameter Table (CPT)

All class-related parameters are stored in the Class Parameter Table. The Class Parameter Table contains the pointer to the Class Queue Descriptor and to the Mux Queue Descriptor.

Table 37-13. Class Parameter Table

Offset	Name	Width	Description
0x00	CMR	word	Class Mode register.See Table 37-14.
0x04	—	Word	Reserved. Should be cleared
0x08	Tx_QD_ptr	Word (Internal)	Transmit MLMC Queue Descriptor Pointer. Points to the Queue Descriptor of the class, which is located in internal RAM and occupies 16 bytes. Should be 16 bytes aligned.
0x0C	—	Word	Reserved. Should be cleared.
0x10	Tx_FSBD_offset	Byte	Fragment Shared Buffer offset. Initialized to 0xff. For RISC use only.
0x11	—	Byte	Reserved. Should be cleared.
0x12	—	Hword	Reserved. Should be cleared.
0x14	Cls_Ext_PTR	Word (internal)	Class Extension pointer. Allocate 32 bytes. Points to a data structure which is an extension to the CPT table. See Table 37-15.
0x18–0x2D	—	22 Bytes	Reserved. Should be cleared.
0x28	—	Word	Reserved. Should be cleared.
0x2E	WBD Size	Hword	Number of entries in the Window BD/Status Ring. The size indicates the number of fragments stored in the WBD/Status ring. This entry must be initialized with a number which is a power of 2 and the value that is written in this entry is 2^n -1. For example: for 128 entries it should be initialized to: 0x7F. See 37.16.1 for details.
0x30	WBD Base	Word	Window BD base pointer. Pointer to external memory.

Table 37-13. Class Parameter Table (continued)

Offset	Name	Width	Description
0x34	Status ring Base	Word (Internal)	Pointer to the Status Ring Table described in 37.6.5.1.2. The table should be aligned to it's size in bytes.
0x38–0x3D	—	6 Bytes	Reserved. Should be cleared
0x3E	Rx_Updating_M_Link	Byte	RISC internal use. Initialize to one of the links ID which are part of the bundle for this class.
0x3F	—	Byte	Reserved. Should be cleared
0x40	Rx LSQN	Word	Should be initialized to 0x01. Internal use. Sequence number of the fragment represented by EF_index.
0x44	Rx Max	Word	Internal use. Initialize to 0x0. The current maximum of the most recently received sequence number over all member links in bundle. This entry is copied by the host when adding a link dynamically. see Section 37.22.2.2, "Addition of Link."
0x48	—	Word	Reserved. Should be cleared.
0x4C	Rx_Queue Descriptor pointer	Word (external or internal)	Points to the Queue descriptor associated to this Class. This queue is for non Mux traffic. Can reside in external or internal memory.
0x50	Rx_Mux Queue Descriptor pointer	Word (external or internal)	Points to the Queue Descriptor used by the De-Mux processing for enqueuing PPP Mux sub-frames. Can reside in external or internal memory.
0x54	RXFRM_CNT	Word	Received frames counter. Initialize to 0.
0x58	Rx FRLCNT	Word	Fragment Loss counter- Counts the number of times window moved due to fragment lost. Initialize to 0.
0x5C	Rx WFCNT	Word	Window Full counter- Counts the number of fragments that were discarded due to lack of space in WBD. Initialize to 0.
0x60	Rx CQBSY_CNT	Word	Class Queue Busy counter: discarded frames counter due to lack of empty BD's in the Core BD Ring. Initialize to 0.
0x64	Rx MRRU_CNT	Word	Longer than MRRU frame counter. Initialize to 0.
0x68	—	Word	Reserved. Should be cleared.
0x6C	DMXBSY_CNT	Word	Demux Busy counter: discarded frames counter due to Busy occurrence in the Demultiplexer Table. Initialize to 0.
0x70	—	Word	Reserved. Should be cleared.
0x74	Rx SQN_TH	Word	Sequence number threshold. Whenever MX exceeds LSQN by more than a threshold value, a fragment loss is detected. See description of threshold register in Section 37.13.1, "Threshold register." See explanation at Section 37.6.5.3, "Fragment Loss Process."
0x78	—	8 Bytes	Reserved. Should be cleared.

37.13.1 Threshold register

This value represents the threshold that will trigger a fragment lose indication due to a big differences between links. It has a different format when working in 12 bit sequence number in the ML PPP or 24 bits for the sequence number.

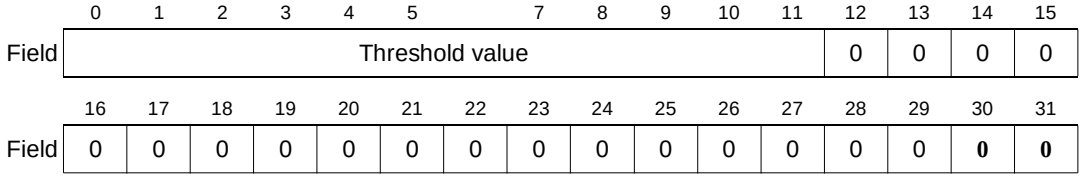


Figure 37-25. Rx Threshold register for 12 bit SN (RX SQN_TH)

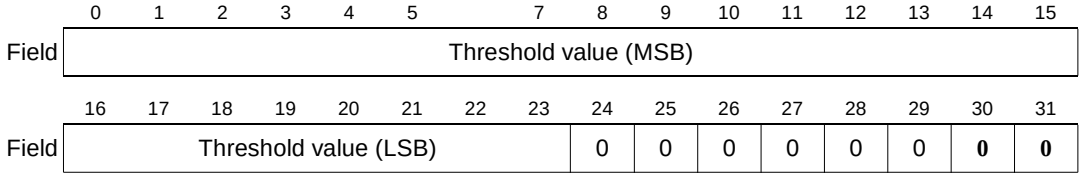


Figure 37-26. Rx Threshold register for 24 bit SN (RX SQN_TH)

37.13.2 Class Mode Register (CMR)

The class mode register (CMR) is a user-initialized register, shown in Figure 37-27.

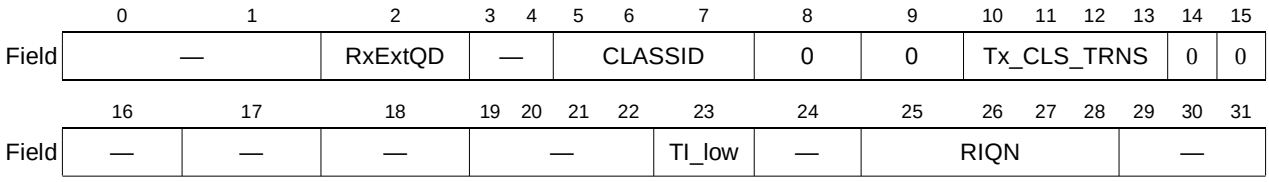


Figure 37-27. Class Mode Register (CMR)

CMR fields are described in Table 37-14.

Table 37-14. CMR Field Descriptions

Bits	Name	Description
0-1	—	Reserved. Should be cleared.
2	RxExtQD	Receiver Queue Descriptors location.(Mux and non Mux) 0 The Queue descriptor is located in the Multi-user RAM 1 The Queue descriptor is located in the external memory
3–4	—	Reserved. Should be cleared.
5–7	CLASSID	Class ID. Assign an ID number for this class, according to the location of the class. Used by the RISC for interrupt information.
8–9	0	Should be cleared.

Table 37-14. CMR Field Descriptions (continued)

Bits	Name	Description
10–13	Tx_CLS_TRNS	Transmit Class Translation. The user should program the CLS bits that will be actually transmitted in the MLMC header. For Short sequence mode: bits 12-13 have to be cleared and bits 10-11 represent the 2 class bits that will be transmitted. For Long sequence mode: bits 10-13 represent the 4 class bits that will be transmitted. Note that there is no obligation that the Tx_CLS_TRNS will match the CPT[ClassID]; for example, the user can assign the CLS=12 (for Long sequence mode) for the class no. 1 (CLASSID=1).
13–15	0	Should be cleared.
16–22	—	Reserved. Should be cleared.
23	TI_low	Transmit Interrupt low event register. Use lower part of MCCE. 0- The interrupts associated with a Multi-Link/Class events resides in queue #5 1- The interrupts associated with a Multi-Link/Class events resides in queue #12 (0xC)
24	—	Reserved. Should be cleared.
25–28	RIQN	Receiver interrupt queue number. Specifies the interrupt queue number on a class level event.- Numbering of the interrupt queues is described in Section 37.20.4, “MCC Event and Mask Registers” which is different then the one of the legacy MCC. Since interrupt queues 4,5 and 11,12 are designated for transmitter usage, the interrupt queues that are available for the receiver are as follows: 0000 Rx Interrupt queue number 0 0001 Rx Interrupt queue number 1 0010 Rx Interrupt queue number 2 0011 Rx Interrupt queue number 3 0110 Rx Interrupt queue number 6 0111 Rx Interrupt queue number 7. 1000 Rx Interrupt queue number 8 1001 Rx Interrupt queue number 9. 1010 Rx Interrupt queue number 10 (0xA) 1101 Rx Interrupt queue number 13 (0xD)
29–31	—	Reserved. Should be cleared.

37.13.3 Class Extension Table

This table is an extension to each CPT table so that each CPT consumes only (0x80) 128 bytes. The size of the extension table is (0x20) 32 bytes

Table 37-15. Class Extension Table

Offset	Name	Width	Description
0x00	Rx_CL0_SQ	Word	Receiver Class Link x Sequence Number: most recently received sequence number over link x of the bundle.See Table 37-16.. This is for the RISC internal use. The Host should initialize these entries with 0.
0x04	Rx_CL1_SQ	Word	
0x08	Rx_CL2_SQ	Word	
0x0C	Rx_CL3_SQ	Word	
0x10	Rx_CL4_SQ	Word	
0x14	Rx_CL5_SQ	Word	
0x18	Rx_CL6_SQ	Word	
0x1C	Rx_CL7_SQ	Word	

37.13.4 Class Link Sequence Number (CLx_SQ)

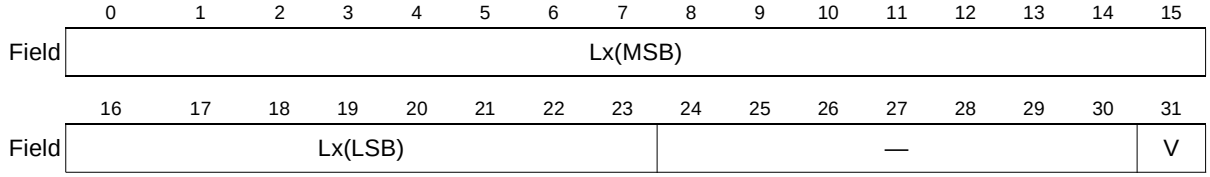


Figure 37-28. Class Link Sequence Number (CLx_SQ)

Table 37-16. CL x_SQ Field Descriptions

Bits	Name	Description
0–23	Lx	Most recently received sequence number over link x of the bundle. Initialize to 0.
24–30	—	Reserved. Should be cleared.
31	V	Valid bit. 0 Link x is not a part of the bundle associated to this class 1 Link x is a part of the bundle associated to this class

37.14 WFQ Table

There are two locations where WFQ tables are used: On the Link level for selection of queue on the link and in the bundle level for class selection. The pointers to the WFQ table are located in LPT(Tx_LWFQ_ptr) and BPT(Tx_CWFQ_ptr).

Table 37-17 describes the WFQ table fields for Link WFQ (LWFQ) or Class WFQ (CWFQ).

Table 37-17. WFQ Table

Offset	Name	Width	Description
0x00	WFT0	Hword	WFTx- Weighted Finish Time. Eight 16 bit FTMs, one for each Queue. The FTM is used to determine the finish time of the next Frame being transmitted from the queue. FTM at offset zero corresponds to Queue0, FTM at offset 0x0E corresponds to Queue7.
0x02	WFT1		
0x04	WFT2		
0x06	WFT3		
0x08	WFT4		
0x0a	WFT5		
0x0c	WFT6		
0x0e	WFT7		
0x10	WINC0	Hword	WINCx- Weighted Increment. Eight 16 bit increments, one for each Queue. INC at offset 0x10 corresponds to Queue0, INC at offset 0x1E corresponds to Queue7.
0x12	WINC1		
0x14	WINC2		
0x16	WINC3		
0x18	WINC4		
0x1a	WINC5		
0x1c	WINC6		
0x1e	WINC7		

37.14.1 WFQ Finish Time Entry (WFT)

Figure 37-29 shows the format of the Weighted Fair Queueing (WFQ) Finish Time entry in the WFQ Table. There are 8 such entries in the table, one for each queue. These entries are initialized by the CPU.

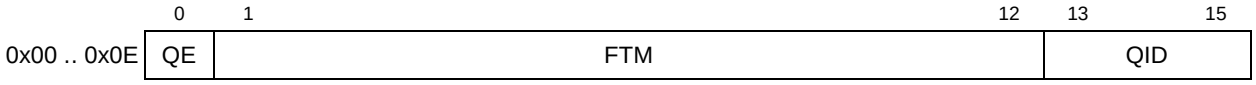


Figure 37-29. WFT Entry

Table 37-18 describes the WFT entry fields.

Table 37-18. WFT Entry Field Descriptions

Bits	Name	Description
0	QE	Queue Empty. Initialized to 1. The CPU clears this bit ONLY if this bit is already set, and after it set the related BD[R] bits of this queue. This bit is set automatically by the RISC if the corresponding Queue has no BD[R] set for this queue. **Note:** If the queue is a MUX queue, this bit will be handheld automatically by the RISC. The host shouldn't change this bit after initialization to 1. The MUX encapsulation process will clear this bit upon completion preparation of a super-frame.
1–12	FTM	Finish time. This bit is automatically updated by the RISC. Initialize with WINC[INC] value.
13–15	QID	Queue ID. The CPU initializes the number of the Queue in these bits. Offset values are as follows: 0000 - 000 0010 - 001 0100 - 010 0110 - 011 1000 - 100 1010 - 101 1100 - 110 1110 - 111

37.14.2 WFQ Increment Entry (WINC)

Figure 37-30 shows the format of the Weighted Fair Queueing (WFQ) Increment entry in the WFQ Table. There are 8 such entries in the table, one for each queue. These entries are initialized by the CPU.

Figure 37-30. WINC Entry

Table 37-19 describes the WINC fields.

Table 37-19. WINC Field Descriptions

Bits	Name	Description
0	—	Reserved. Should be cleared.
1–12	INC	Weighted Increment value. This value determines the Priority for the queues. If all queues are backlogged the Priority of each queue is 1/INC. If some of the queues do not have something to transmit, the total Priority of the port is divided among the backlogged queues according to their relative INC values. INC=0 for all 8 queues - Queue 0 has the highest priority (Fixed Priority). See Example 37.7.3.4.
13–15	—	Reserved. Should be cleared.

37.15 Queue Descriptors (QD)

37.15.1 RX Queue Descriptor

The RX Queue Descriptor is a table which describes the BD Ring associated to a Class or to a Link.

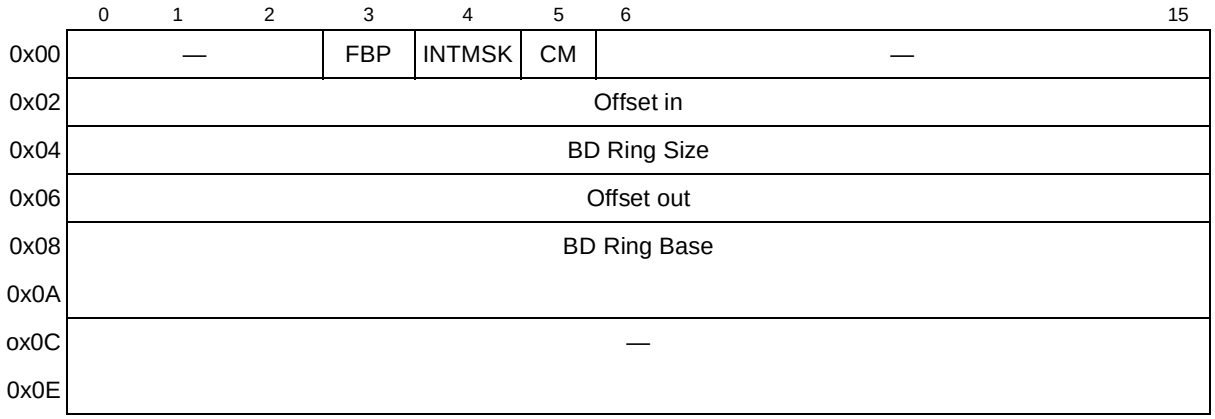


Figure 37-31. RX Queue Descriptor

Table 37-20. RX Queue Descriptor Fields

Offset	Bit	Name	Description
0x00	0–2	—	Reserved. Should be cleared.
	3	FBP	Free Buffer Pool Queue. 0 The BD's of this queue contain Data pointers which are swapped upon reception with the new data pointers. 1 The BD's of this queue do not contain any Data pointer. The Data pointers are located in the Free buffer pool associated to this RQD. For PPP Mux queues and LCP this bit is not valid. **Note:** Mux operation always requires FBP and LCP frames are always using swap operation.
	4	INTMSK	Receive frame mask. 0 Disable interrupt related to this queue (RXF for non Mux, LE and FRG for muxed queue). 1 Enable interrupt related to this queue (RXF for Non Mux, LE and FRG for muxed queue).
	5	CM	Continuous mode 0 Normal operation. 1 The RISC does not clear the ready bit after this BD is closed.
	6–15	—	Reserved. Should be cleared.
0x02	0–15	offset IN	Points to the next BD in the BD ring. Initialize to 0
0x04	0–15	BD Ring Size	Number of BD's in the Ring.
0x06	0–15	Offset out	Pointer to the next BD to be fetched by the Host. Updated by the host. Initialize to zero.

Table 37-20. RX Queue Descriptor Fields (continued)

Offset	Bit	Name	Description
0x08	0–31	BD Ring Base	BD Base Address. Address of first BD in the queue.
0x0C	0–31	—	Reserved. Should be cleared

37.15.2 TX Queue Descriptor

As shown in Figure 37-3, there are up to 8 Link queues under each link. Also there are optionally up to eight ML queues, one for each class. Each queue has its priority level, which is described in the WFQ section.

The TX Queue Descriptor (TQD) length is 16 bytes. For the Link queues, it could reside in internal or external memory. For the ML queues, it is located in the internal memory. The QD contains the information about the current BD handled. The BD base address and BD currently being transmitted. It also contains mode bits for selecting the mode of operation.

For Muxed TQD the RISC needs additional parameters for proper operation. This information, such as User Defined Header (UDH), resides in the PPP Mux Parameter Table.

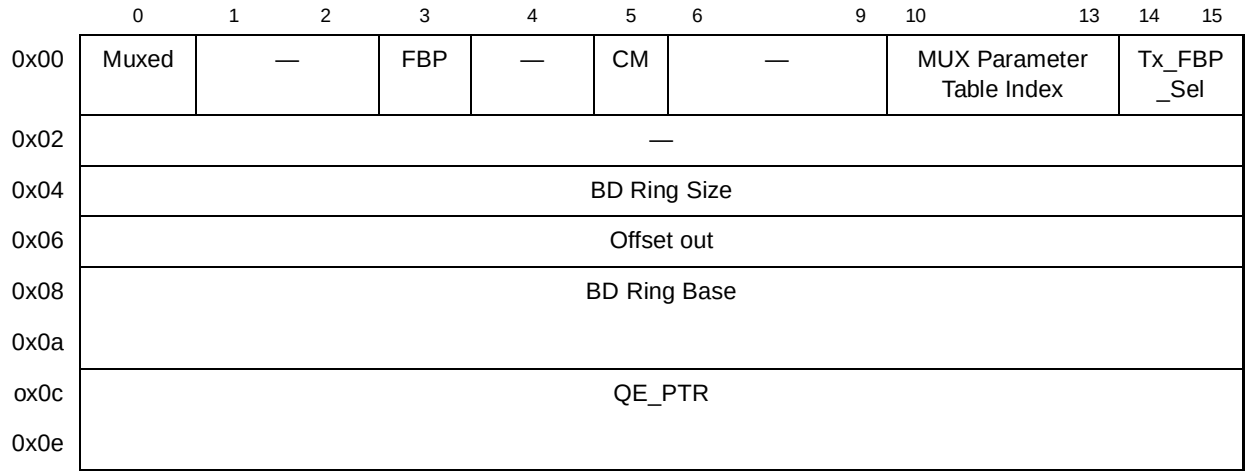


Figure 37-32. TX Queue Descriptor

Table 37-21 describes the TQD fields.

Table 37-21. TX Queue Descriptor Fields

Offset	Bit	Name	Description
0x00	0	Muxed	PPP Multiplex Mode. 0 - PPP Multiplexing is disabled for this QD. 1 - PPP Multiplexing is enabled for this QD.
	1–2	—	Reserved. Should be cleared.
	3	FBP	Free Buffer Pool Mode enabled. Important: This mode is valid for link queues only. When the queue is a ML/MC queue FBP operation is always needed regardless of the setting of this bit. 0 Data pointers in the BDs are not returned to the FBP after transmission 1 Data pointers are returned to the FBP after transmission.
	4	—	Reserved. Should be cleared.
	5	CM	Continuous mode 0 Normal operation. 1 The RISC does not clear the ready bit after this BD is closed.
	6–9	—	Reserved. Should be cleared.
	10–13	MUX Parameter Table Index	Index of the Mux encapsulation parameter table which is associated with this queue. see 37.19.6
	14–15	Tx_FBP_Sel	Selects one of 4 available Free Buffer Pools. The actual address to the FBP table is BPT(FBPT_BASE)+BPOOL*16. See Section 37.17.2, “TX Free Buffer Pool.”
0x02	0–15	—	Reserved. Should be cleared
0x04	0–15	BD Ring Size	Number of BD’s in the Ring.
0x06	0–15	Offset out	Pointer to the next BD to be fetched by the RISC. Updated by the RISC. Initialize to zero.
0x08	0–31	BD Ring Base	BD Base Address. Address of first BD in the queue.
0x0C	0–31	QE_PTR	This is a pointer to the queue Empty /Class Empty bit depending on the setup. The address of this bit is specified in 0xQueue Empty Bit Management

37.16 Receive Window BD (WBD) Ring and Status Ring

In the process of receiving ML/MC traffic each class has a Status ring and a WBD ring associated with it. This status ring is located in internal memory and each entry of it represents a WBD entry from the ring which resides in external memory. All these structures are for RISC internal use and the application should allocate the memory required for each data structure. Each status ring entry is 4 bits width (A nibble) and the WBD entry is 16 bytes long.

NOTE

When working in the free buffer pool mode (QD[FBP]=1), the status ring table has to be initialized to ALL zeros at the beginning of ML/MC operation. If QD[FBP]=0, each entry in the status ring table has to be initialized to 0x4 (this bit indicates full buffer). By doing so the RISC swaps the data pointers from the WBD to the BD ring and vice versa. In this case, all the WBD should naturally be initialized with valid data pointers.

The WBD ring described below is for information only and the application does not have to initialize or handle any of it.

Figure 37-33 shows an entry of the WBD ring.

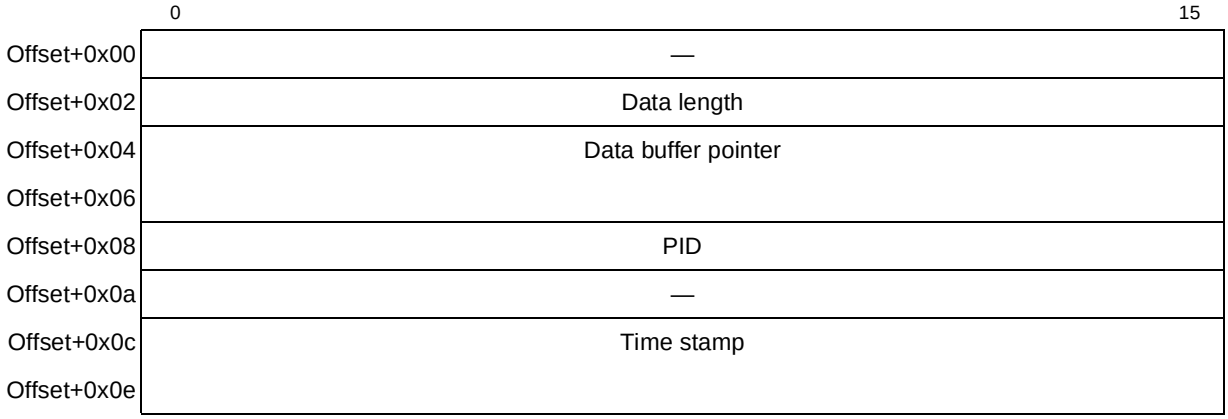


Figure 37-33. WBD Entry

Table 37-22 describes the WBD entry fields.

Table 37-22. WBD Entry Fields

Offset	Bit	Name	Description
0x00	0–15	—	Reserved
0x02	0–15	Data length	Length of the Data received in this fragment Data buffer.
0x04 0x06	0–15	Data buffer pointer	Need to be initialized only if RxQD[FBP] is cleared. Points to the data buffer associated with this WBD entry. In this case each entry in the StatusRing table has to be initialized with 0x4.
0x08	0–15	PID	Protocol ID associated with this WBD entry. Valid only for an entry marked B in the Status Ring.
0x0A	0–15	—	Reserved
0x0C 0x0E	0–15	Time stamp	Time Stamp copied from the QUICC Engine Time-Stamp Register (CETSR) after receiving the complete fragment, see Section 20.3.10, “QUICC Engine Time-Stamp Registers (CETSRn).”

37.16.1 WBD Ring and Status Ring Size

The user must allocate a sufficient place for the WBD ring according to the requirements of the system. The minimum size of the WBD ring has to contain at least a whole packet. In addition, the differential delay between 2 links in a same bundle has to be taken into consideration. This size should be allocated per class on the ML/MC system. Also, the size of the WBD should have enough entries to contain at least one complete PPP packet with the size of MRRU. This defines a minimum size for Window BD. Table 37-23 defines the variables needed to calculate ring size.

Table 37-23. WBD Ring Size Equation Variables

Variable	Definition
Dt	Differential delay between links [seconds]
N	Number of links in the bundle
Ri	Rate of link i [bps]
MRU	Fragment size [bytes]

Thus, the number of entries in the WBD Ring is calculated as follows:

$$\mathrm{Max}\left(\frac{MRRU}{MRU}, \sum_{i}^{N-1} \frac{R_i}{MRU \times 8} \times Dt\right) \times 16$$

This result should be rounded up to the next 2^N value.

Size of WBD in bytes is 2^N*16 [bytes].

Size of Status Ring = (2^N/2) [bytes].

37.16.2 Rx Packets Reconstruction Management

The Packets Reconstruction Management is placed in the MURAM. Table 37-24 describes the Packets Reconstruction Management. This structure size is 64 bytes and it should be 64 bytes aligned.

Table 37-24. Packets Reconstruction Management

Offset	Name	Width	Description
0x00	—	24 Bytes	Reserved, should be cleared.
0x18	Packet Reconstruction Table Base	Word (internal)	Base of the Packet Reconstruction Table (PRT). The size of this table is determined by the application. Should be word aligned. **Note:** When ML MC process is not needed on the receive side, but PPP Mux frames are received on the link, the user still needs to define this base, but only one Rx packet reconstructing entry is needed in this case (this entry in the table should be initialized to 0xc000).
0x1C	—	36 Bytes	Reserved, should be cleared.

37.16.2.1 Rx Packets Reconstruction Entry

The size of the table is determined by the application and last entry is marked with a Wrap indication. Every entry represents a frame. An entry is described in Figure 37-34.

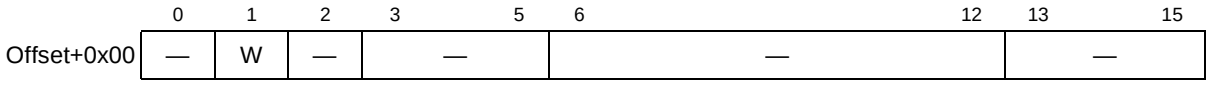


Figure 37-34. Packets Reconstruction Table Entry

Table 37-25. Packets Reconstruction Table Entry Fields

Offset	Bit	Name	Description
0x00	0	—	Must be initialized to 1.
	1	W	Wrap bit. During initialization, the host must set the W bit only for the last entry in the circular table. All other entries should have this bit cleared.
	2–15	—	Reserved, should be cleared.

37.17 Free Buffer Pools (FBP)

37.17.1 Rx Free Buffer Pool

Free buffer Pool structure is located in external memory. It is an array of data pointers as described in Figure 37-35 The buffer Pool has a management table associated with it.

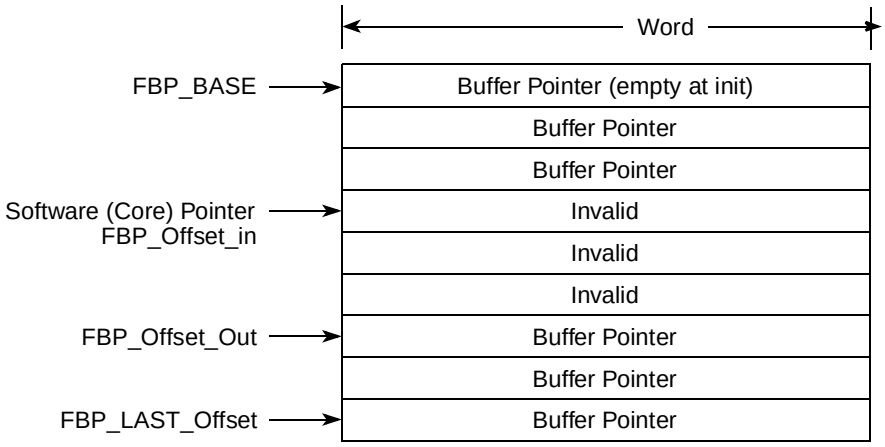


Figure 37-35. Rx Free Buffer Pool per Bundle.

Figure 37-36 describes the Free Buffer Pool Parameter Table. The host should maintain the Offset_In parameter in this table so that the RISC can calculate availability of buffers in the pool.

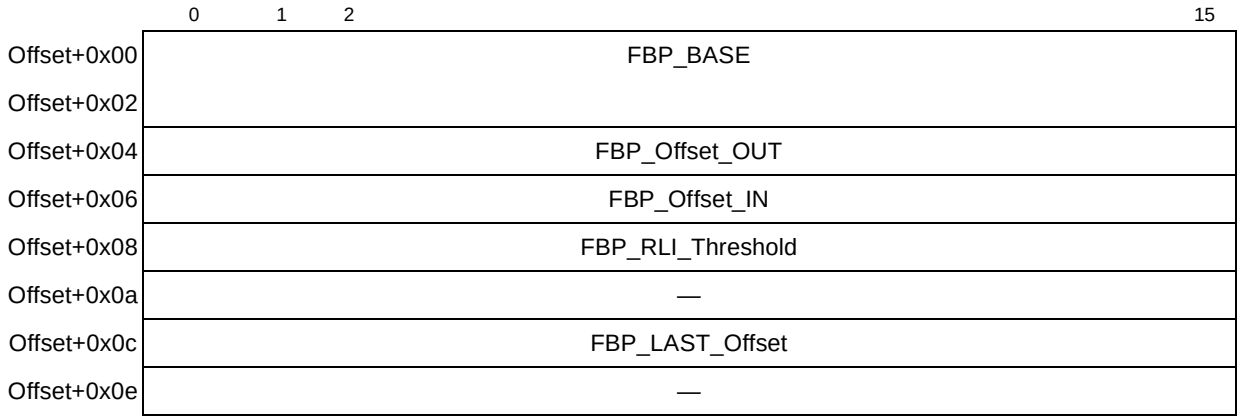


Figure 37-36. RX Free Buffer Pool Parameter Table

The user prepares a BD ring without assigning buffers to them (no Data buffer pointer). The user also prepares a set of free buffers (of size: max LPT[MRU]) in a free buffer pool.

Table 37-26 describes the RX Free Buffer Pool Parameter Table fields.

Table 37-26. RX Free Buffer Pool Parameter Table Fields

Offset	Bit	Name	Description
0x00 - 0x02	0–31	FBP_BASE	Free buffer pool base. Holds the pointer to the first entry in the free buffer pool. FBP_BASE should be word aligned. User-defined.
0x04	0–15	FBP_Offset_OUT	Free buffer pool Receiver pointer. Offset from the FBP_BASE to the current entry used by the receiver. The actual address of the current received FBP entry = FBP_BASE + FBP_Offset_OUT. For termination mode **Initialize to 0x04.** When the Init command is issued the RISC will take the buffer pointer pointed by this entry and will advance the offset to the next entry.
0x06	0–15	FBP_Offset_IN	Free buffer pool host pointer. Offset from the FBP_BASE to the current entry which is being returned to the free buffer pool. **Initialize to 0x0.** This implies this entry is empty and available for a returned pointer. In termination mode The host should maintain this offset **AFTER** each buffer pointer is returned to the pool. **IMPORTANT**: For proper synchronization between the receiver and the host, the host should not advance this entry so it would match the offset out value. (The RISC will advance it's offset_out value to match offset in and in this case the FBP is fully exhausted a BP_BSY interrupt is issued by the RISC)
0x08	0–15	FBP_RLI_Threshold	When the difference between the Offset_In and Offset_Out has reached this value an interrupt is generated. It indicates that the amount of buffers left in the system has reached the low watermark. If the values programmed in this field is 0x0 no interrupt is issued.
0x0A	0–15	—	Reserved. Should be cleared.
0x0C	0–15	FBP_LAST_Offset	Free buffer pool last pointer. User initialized to the offset of the last entry in the FBP. The actual address of the last FBP entry = FBP_BASE + FBP_LAST_Offset.
0x0E	0–15	—	Reserved. Should be cleared.

Each entry of the Free Buffer Pool is located in the external memory.

37.17.2 TX Free Buffer Pool

Four pools are available for the transmitter. The structure for free buffer pools is located in external memory, and is identical to this of the receiver As described in Figure 37-37. The purpose for this is to enable better memory utilization so different queues could use different sizes of buffers.

Important: When working in ML/MC queues, Free Buffer Pool operation is mandatory and the user should initialize the parameters and data structures for this mode.

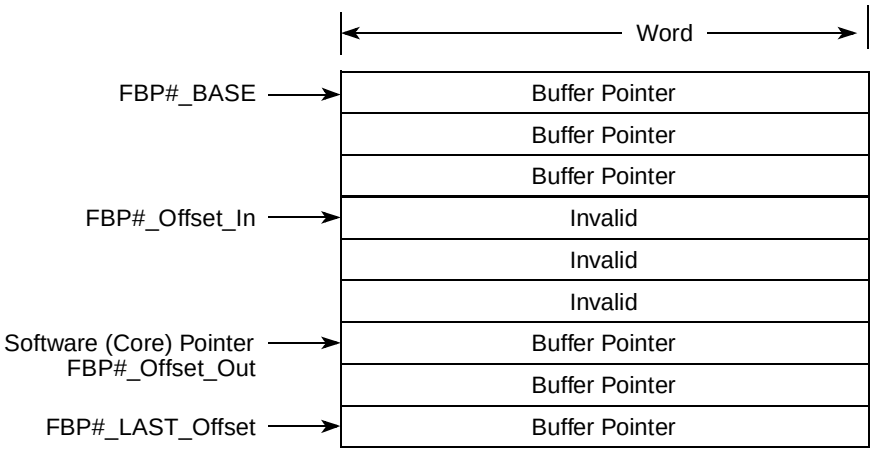


Figure 37-37. Free Buffer Pool Structure

37.17.3 Tx Free Buffer Pool Parameter Tables

The free buffer pool parameters are held in parameter tables in the multi-user RAM; FBPT_BASE in the BPT 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 FBPT_BASE+ QD[BPOOL] * 16.

NOTE

FBP is empty at initialization and is being filled by the QE upon completion transmitting of each buffer. Figure 37-38 describes the Free Buffer Pool Parameter Table.

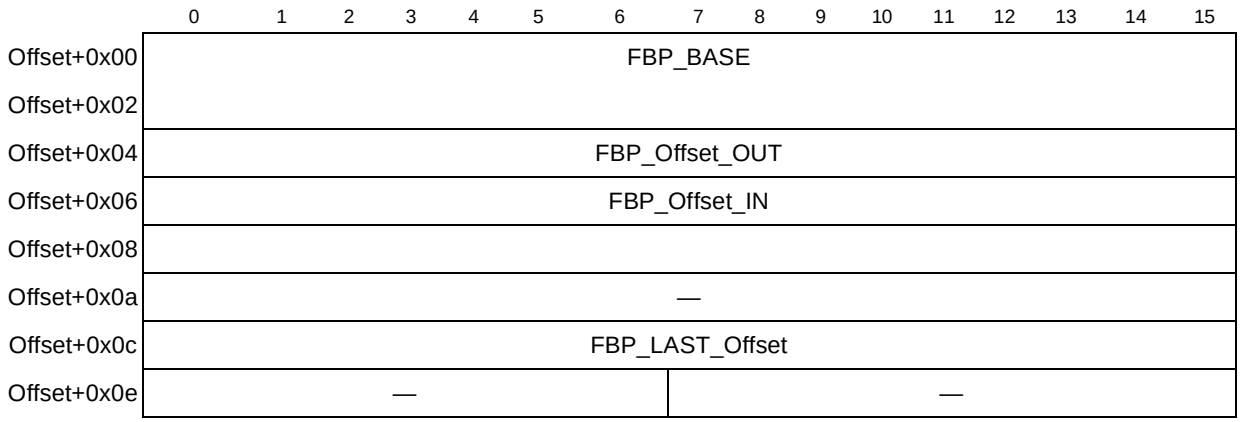


Figure 37-38. TX Free Buffer Pool Parameter Table

Table 37-27. Tx Free Buffer Pool Parameter Table

Offset [1]	Bits	Name	Description
0x00	0–31	FBP_BASE	Free buffer pool base. Holds the pointer to the first entry in the free buffer pool. FBP_BASE should be word aligned. User-defined.
0x04	0–15	FBP_Offset_OUT	Host offset out indicates the pointer entry which the host is currently returning to the application after transmission has finished. The actual address of the current received FBP entry = FBP_BASE + FBP_Offset_OUT. For termination mode **Initialize to 0x00**. The host should maintain this entry in this data structure in order to keep proper buffer management.
0x06	0–15	FBP_Offset_IN	Free buffer pool transmit pointer. Offset from the FBP_BASE to the current entry in the free buffer pool to which the transmitter will return the buffer pointer after it's transmission. *Important*: The transmitter will return a buffer to this offset if the difference between this value and the Offset_Out value is greater than one. The actual address of the current transmitted FBP entry = FBP_BASE + FBP_Offset_IN. For termination mode **Initialize to 0x00**.
0x08	0–15	—	Reserved. Should be cleared.
0x0A	0-15	—	Reserved. Should be cleared.
0x0C	0–15	FBP_LAST_Offset	Free buffer pool last pointer. User initialized to the offset of the last entry in the FBP. The actual address of the last FBP entry = FBP_BASE + FBP_LAST_Offset.
0x0E	0–15	—	Reserved. Should be cleared.

[1] Offset from FBPT_BASE+TQ[BPOOL] × 16

the application should perform the following sequence when getting a buffer pointer from the buffer pool:

-Read offset out value.

-read the buffer pointer located in this offset in the buffer pool. Check if the pointer value is different than zero. If so the pointer is valid and can be reused for transmission again. If it is zero perform a second read of the entry until a non zero value is read.

-After reading a valid pointer the application should clear the entry in the buffer pool. (so on the next time it will be used as indication that a new buffer has been placed by the QE)

37.18 Buffer Descriptors (BD)

37.18.1 Receive Buffer Descriptor (RxBD)

Figure 37-39 shows the RxBD.

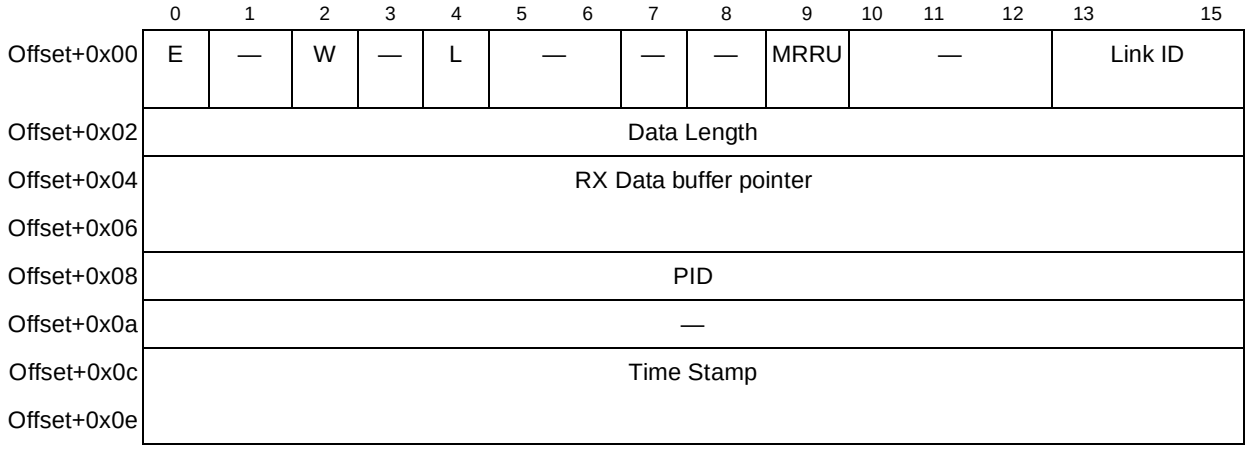


Figure 37-39. Receive Buffer Descriptor

Table 37-28 describes the RxBD fields.

Table 37-28. Receive Buffer Descriptor Fields

Offset	Bit	Name	Description
0x00	0	E	Empty bit. 0 The buffer associated with this RxBD is filled with data. The core can examine or write to this BD. The RISC does not use this BD again while E remains zero. 1 The RX buffer is empty or enqueuing is in progress.
	1	—	Reserved.Should be cleared.
	2	W	Wrap bit 0 This is not the last BD in the RxBD ring. 1 This is the last BD in the RxBD ring.
	3	—	Reserved. Should be cleared.
	4	L	Last. 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
	5–10	—	Reserved. Should be cleared.
	9	MRRU	MRRU exceed. This bit is set by the RISC in the last BD of a frame to indicate that the frame length exceeds MRRU value.
	10-12	—	Reserved. Should be cleared.
	13–15	Link ID	This field is valid only for plain PPP, plain PPP mux & LCP queues. It indicates on which Link the data was received. It is useful when several links use the same queue.
0x02	0–15	Data Length	The data length is the number of bytes stored in this BD's data buffer. If L bit is set this field contains the length of the complete frame.
0x04–0x06	0–31	Rx Data buffer Pointer	Points to the location of the associated buffer. If QD[FBP] = 0, this field must be initialized with a data buffer pointer. **Note:** IF QD[FBP] = 1, this field is ignored by the RISC.
0x08	0–15	PID	PPP PID value of the PPP frame. This entry is valid for the first BD of each PPP frame so that for MLMC PPP frames which takes several BDs only the first BD is valid.
0x0A	0–15	—	Reserved. Should be cleared.
0x0C	0–31	Time Stamp	The RX copies the Time Stamp from QUICC Engine Time-Stamp Register (CETSR) after receiving a complete frame/fragment, see Section 20.3.10, "QUICC Engine Time-Stamp Registers (CETSRn)."

37.18.2 Receive LCP Buffer Descriptor (RxBD)

LCP packets have a dedicated queue and the BDs associated with the queue do not contain the extra information provided by a ML/MC BD. Therefore the size of the BD is smaller.

As described in Section 37.1, "Overview," the LCP BD ring is initialized with valid data pointers and once an LCP packet is received the data pointer which is currently in the link parameter table is swapped with the pointer of the BD currently used by the receiver.

Figure 37-40 shows the RxBD. All the information is identical to the Receive BD.

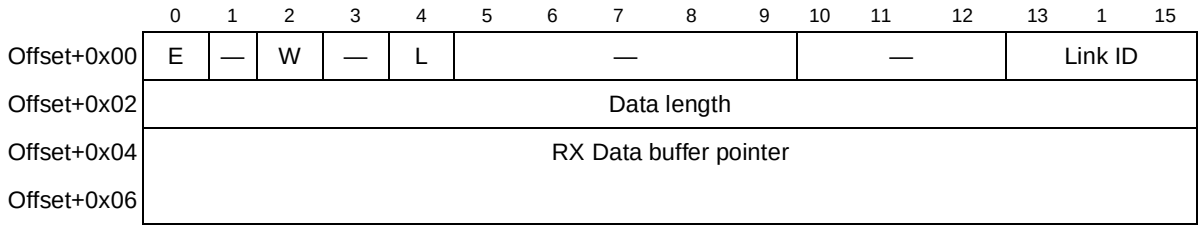


Figure 37-40. Receive LCP Buffer Descriptor

37.18.3 Transmit Buffer Descriptor (TxBD)

Figure 37-41 shows the TxBD.

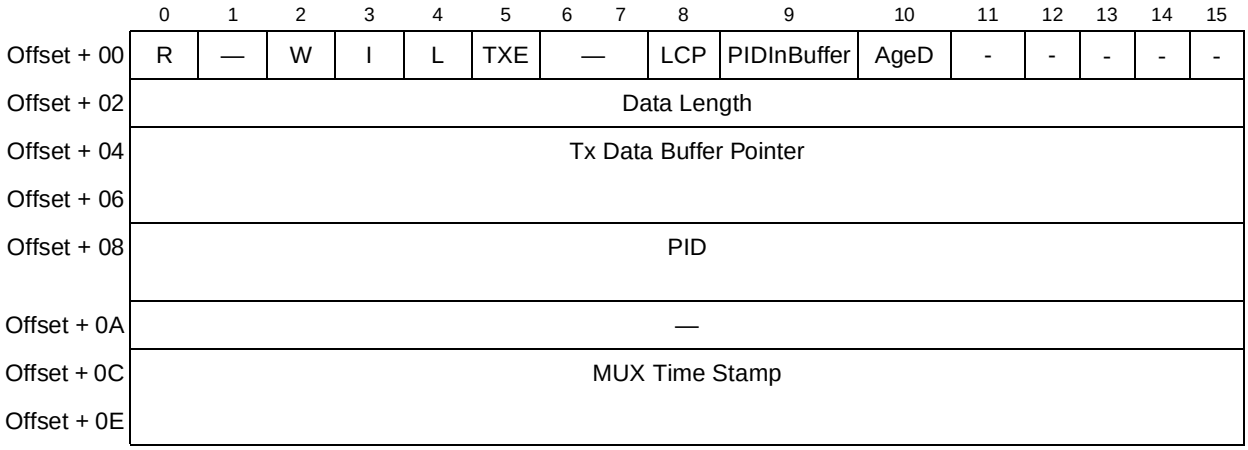


Figure 37-41. Transmit Buffer Descriptor (TxBD)

The TxBD length is 16 bytes. Table 37-29 describes the TxBD fields.

Table 37-29. TxBD Field Descriptions

Offset	Bits	Name	Description Weigthed Fair Queueing (WFQ)
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 data buffer. The RISC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer is ready to be transmitted. The transmission may have begun, but it has not completed. The user cannot modify this BD 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 is used, the RISC receives incoming data into the first BD in the table (the BD pointed to by TBASE). The number of TxBD's in this table is programmable and is determined the wrap bit.
	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 this buffer has been serviced by the MCC. This bit can cause an interrupt (if enabled).
	4	L	Indicates last BD for the frame. For sub-frames indicate last BD of sub-frame. 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
	5	TXE	Transmit Error indication. Set by the RISC under some error conditions which prevent this BD from being transmitted. Valid in PPP MUX sub frames only. An example for such a case is when a sub-frame length exceeds the MAX Superframe size.
	6–7	—	Reserved. Should be cleared.
	8	LCP	LCP frame. Could be set only for TxBD under a Link queue. LCP packet can split over several BD's. The LCP bit should be marked at the first BD containing the LCP packet. The ACFC characters and the PID are always uncompressed.
	9	PIDInBuffer	PID In Buffer. For PPP MUX Frames the PIDInBuffer must be reset. 0 - The PID is located in the BD. 1 - The PID is located in the buffer.
	10	AgeD	Age Discard. The RISC set this bit if the current BD was not transmitted since it was too old. See AgeEn bit, Section 37.19.7, "Mux Mode Register (MMR)." This is valid for MUX subframes BD's only. Should be cleared by the host.
	11–15	—	Reserved. Should be cleared.
0x02	Hword	Data Length	The data length is the number of bytes that should be transmitted from this BD's data buffer. It is never modified by the RISC. The value of this field should be greater than zero.
0x04	Word	Data 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 RISC.
0x08	Hword	PID	PPP PID value. PID value for plain PPP frames, or subframe PID in PPP MUX frames or ML/MC frames.

Table 37-29. TxBD Field Descriptions (continued)

Offset	Bits	Name	Description Weigthed Fair Queueing (WFQ)
0x0A	Hword	—	Reserved. Should be cleared.
0x0C	Word	MUX Time Stamp	Mux Time Stamp. The time stamp in which the user had set the Ready bit of this buffer. The user initialize this field with the value written in the QUICC Engine Time-Stamp Register (CETSR), see Section 20.3.10, "QUICC Engine Time-Stamp Registers (CETSRn)." This Time stamp with the SubFrame Age Period parameter, are used by the RISC, in order to determine if the SubFrame packet is too old. In case when the current Time Stamp (when the packet is intended to be transmitted) minus the MUX Time Stamp result is bigger than the SubFrame Age Period, then the packet is discarded.

37.19 PPP Mux Data Structures

The Mux and Demux processes have a page of their own in the parameter RAM.

37.19.1 PPP Mux Process Parameter RAM

The Parameter RAM holds two pointers that points to the management table of the Demux handler and to the Mux handler. The pointers are located in the internal MCC Global Parameter RAM, see at Table 37-5.

37.19.2 Demux Management Table (DMT)

The Demux Management Table (DMT) contains all the parameters needed for the de-muxing process. It resides in internal memory and it's size is 64 bytes.

Note: In systems where the PPP Mux traffic is received on the link level ONLY i.e. not over the ML/MC, it is still required to configure the Packet Reconstruction parameters (management and table) although there is no need for this functionality in the absence of ML/MC traffic.

	0 1	2	3 4	5	6	7	8 15
Offset+0x00	—	GBL	—	CETM	—	BIB	— IQOV
Offset+0x02	Last entry offset						
Offset+0x04	External DFT Base pointer						
Offset+0x06							
Offset+0x08	Internal Buffer Pointer						
Offset+0x0A							
Offset+0x0C	Offset in (for receiver)						
Offset+0x0E	Offset out (for DeMux)						
Offset+0x10-0x1F	Reserved						
Offset+0x20	DeMux Interrupt queue base (DIQ)						
Offset+0x22							
Offset+0x24	Demux Interrupt pointer (DIP)						
Offset+0x26							
Offset+0x28	—						
Offset+0x2a							
Offset+0x2c							
Offset+0x2e							
Offset+0x30	DInt_THR						
Offset+0x32	DInt_CNT						
Offset+0x34-0x3e	Reserved						

Figure 37-42. DeMux Management Table (Internal Structure)

Table 37-30 describes the DMT fields.

Table 37-30. DeMux Management Table Field Description

Offset	Bits	Field	Description
0x00	0–1	Reserved	Reserved. Should be cleared.
	2	GBL	Global. Setting GBL enables snooping of data buffers, BD's, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "Serial DMA Mode Register (SDMR)."
	3–4	Reserved	Reserved. Should be cleared.
	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 19-1 and Section 5.4.3.7, "Debug Configuration."
	6	Reserved	Reserved. Should be cleared.
	7	BIB	DeMux Fragment Table bus location. 0 - Reside on the coherent system bus 1 - Reside on the secondary bus
	8–14	Reserved	Reserved. Should be cleared.
	15	IQOV	IQOV Indicates the event of an Interrupt queue Overflow When there is a De-Mux event set in the CEVTER register for the de-Mux process (Based on its selected SNUM), the application should read this status bit which indicates a situation where there is no valid entry in the interrupt queue. The host should never reset this bit. It will be cleared by the QUICC Engine block when the full condition of the interrupt queue is gone.
0x02	Hword	Last entry offset	This field contains the offset to the last entry in the DFT table. It is calculated by 32 * (Number of DFT entries -1).
0x04	Word	External DFT Base pointer	This field holds the pointer to the DeMux Fragment Table (DFT).
0x08	Word	Internal Buffer Pointer (Internal)	Points to a 48 bytes space in Multi-user RAM for temporary storage location. r
0x0C	Hword	Offset in (for receiver)	Internal use. Initialized to zero.
0x0E	Hword	Offset out (for DeMux)	Internal use. Initialized to zero.
0x10–0x1F	16 bytes	Reserved	Reserved. RISC internal use.
0x20	Word	DeMux Interrupt queue base (DIQ)	This field holds the pointer to the interrupt queue base. The interrupt queue is dedicated to the DeMux interrupts. Pointer to external memory.
0x24	Word	DeMux Interrupt pointer (DIP)	Pointer to the DeMux Interrupt queue. The RISC writes the next interrupt information to this entry when an exception occurs. Initialize to DeMux Interrupt queue base pointer. Pointer to external memory.

Table 37-30. DeMux Management Table Field Description (continued)

Offset	Bits	Field	Description
0x28–0x2F	8 Bytes	—	Reserved, should be initialized to zero.
0x30	Hword	DInt_THR	DeMux fragment receive Interrupt threshold.
0x32	Hword	DInt_CNT	DeMux fragment receive count. Internal use. Should be initialize to DInt_THR
0x34–0x3F	12 bytes	Reserved	Reserved. RISC internal use.

37.19.3 Demux Fragment Table Entry (DFT Entry)

The Demux Fragment Table (DFT) is a data structure which defines the interface between receiver, the host and the demux process . It resides in external memory. Each entry of the table is associated with a PPP DeMux frame/fragment and it's parameters are being stored in the DFT entries for further process by the demux. Figure 37-43 describes a DFT entry. User should allocate 32 bytes for this table entry.

NOTE

The number of entries in this table should be large enough to accommodate the largest Super-frame that could possibly arrive on the system (i.e. the number of fragments of the largest super-frame in the system).

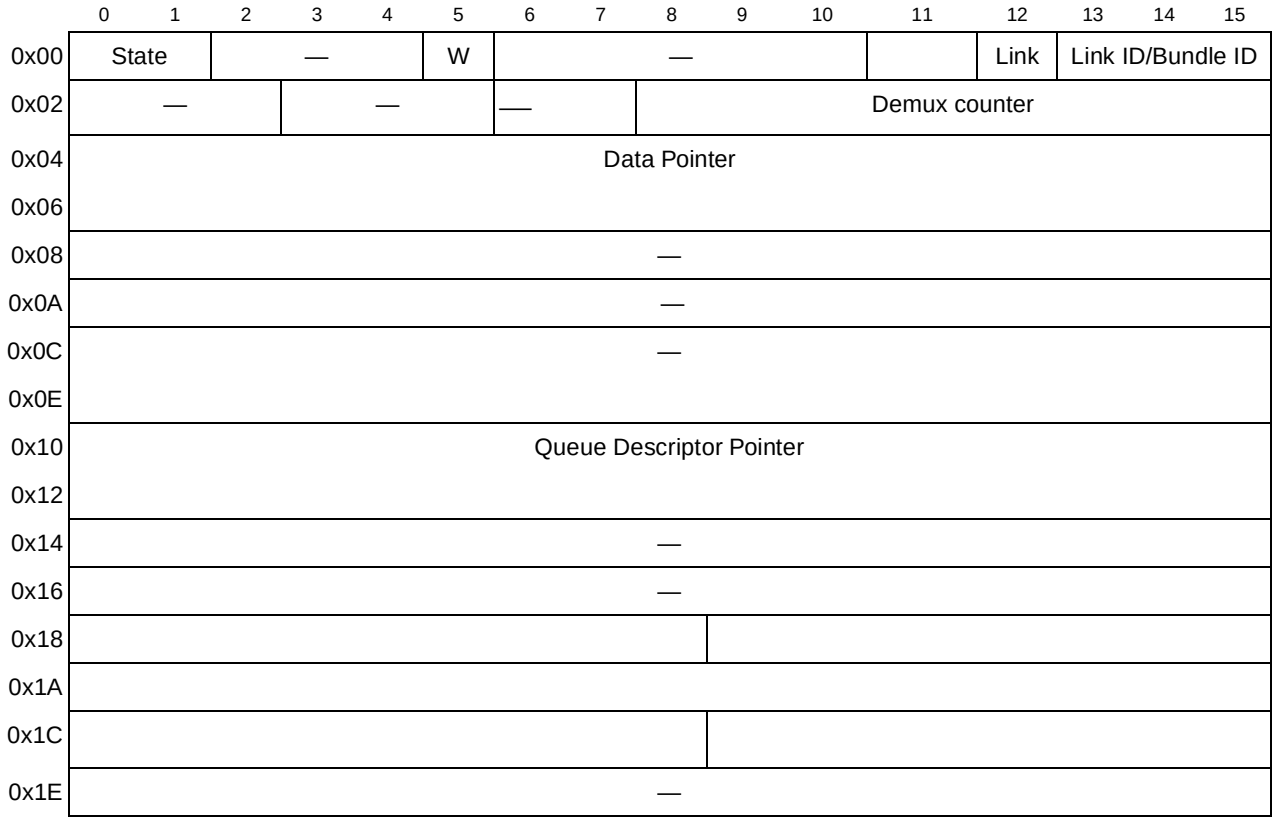


Figure 37-43. DeMux Fragment Table Entry

Table 37-31 describes the Demux Fragment Table entry fields.

Table 37-31. Demux Fragment Table Entry Field Descriptions

Offset	Bits	Field	Description
0x00	0–1	State	00 - This entry is available. The receiver may handle this entry 01 -This entry is now being handled by the DeMux 10 - This entry is now ready for host processing. Once the host finished processing all the SFBD's related to this entry, set the state to 00. 11 - Reserved
	2–4	—	Reserved. Initialized to 0.
	5	W	The Wrap bit indicates the last entry in the DeMux entry table.
	6–11	—	Reserved. Should be cleared.
	12	Link	Initialized to 0. Indicates the host that this entry has been received on a link level and the Link ID information is passed to the BD. 1- Next 3 bits (13-15) indicates the link ID on which this frame has been received. 0- Next 3 bits (13-15) indicates the bundle ID on which this fragment has been received (for ML/MC mode).
	13–15	Link ID/ Bundle ID	Initialized to 0. Link ID from which this fragment has been received. This is valid only for a PPP Mux on a link level and is useful if all links in the bundle use the same DeMux Queue Descriptor i.e. aggregating all the Mux traffic into a single queue. Bundle ID- When the fragment is part of a ML traffic this field indicates the Bundle ID to which this fragment belongs.
0x02			
	0-7	Res	reserved for QUICC Engine block usage
	7-15	Demux counter	Initialized to 0. This field holds the counter of the number of BDs that were processed or are currently processed by the demultiplexer. Once the demux process changes the state field to 0x10 the host has the information of how many sub-frames there are in the specified fragment so it knows how many BD's should be processed before returning the buffer pointer to the FBP. It is used by the host to know how many subframes to process in the superframe, before changing the state from '10' to '00'.
0x04	0–31	Data Pointer	Initialized to 0. Fragment/PPPMux frame data buffer pointer. This data pointer is the "original" pointer from the FBP and should be returned by the host after processing all its sub-frames.
0x08– 0x0F	—	—	Reserved. Initialized to 0.
0x10	0–31	Queue Descriptor Pointer	This field holds the pointer to the Queue Descriptor that handles the BD's to which the subframe of the currently processed fragment should be assigned.
0x14 0x1F	—	—	Reserved. Initialized to 0.

37.19.4 Sub-Frame Rx Buffer Descriptor

The Sub-Frame Buffer Descriptor (SFBD) contains the sub-frame parameters for the host. The data pointer point to the location of the information of sub-frame. This pointer actually points into the original buffer

on which the fragment was stored. No data is being copied so the SFBD do not have any data buffers allocated for their usage. The PID information is extracted and placed in the BD. Figure 37-44 describes the various fields of the SFBD.

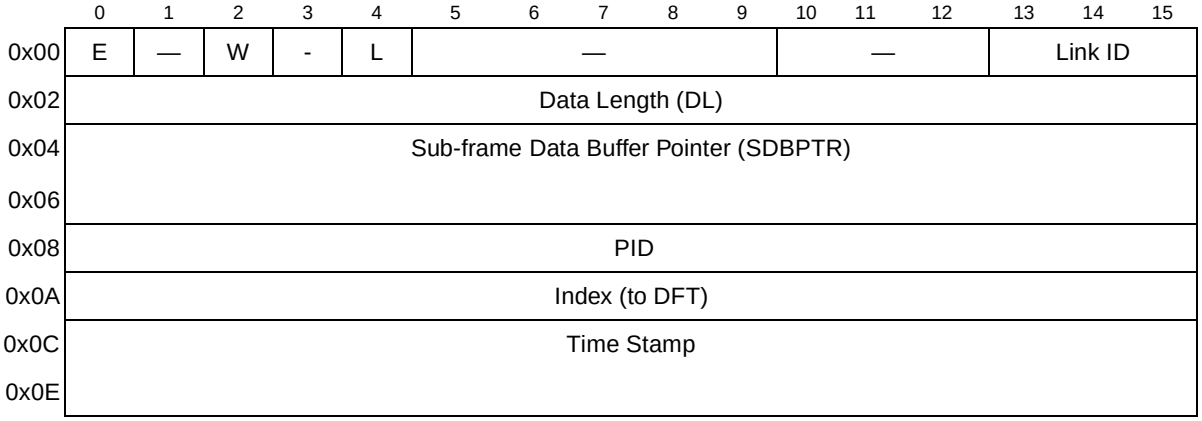


Figure 37-44. Rx Sub-Frame Buffer Descriptor

Table 37-32 describes Sub-frame Buffer Descriptor fields.

Table 37-32. Sub-Frame Buffer Descriptor Field Descriptions

Offset	Bits	Field	Description
0x00	0	E	Empty bit. 0 The buffer associated with this RxBD is filled with data. The core can examine or write to this BD. The RISC does not use this BD again while E remains zero. 1 The RX buffer is empty or enqueuing is in progress.
	1	—	Reserved. Should be cleared.
	2	W	Wrap (final BD in Ring) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the host RxBD ring of this current queue. After this buffer has been used, the RISC receives incoming data for this queue into the first BD in the ring. The number of RxBD's in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	-	Reserved.Should be cleared.
	4	L	Last bit. 0-this BD is not the last BD of the sub-frame. 1-this BD is the last BD of the sub-frame.
	5–12	—	Reserved.Should be cleared.
	13–15	Link ID	This indicates the link ID on which this PPP Sub-frame has been received. It is valid only for MUX superframe received in a Link muxed queue. In ML/MC queue this field is cleared.
0x02	Hword	Data Length (DL)	Contains the length of the buffer associated with this BD. This is actually the length of pure data of the packet. All header information is removed.
0x04	Word	SDBPTR	Sub-frame data buffer pointer. Points to the address of the subframe in the data buffer.
0x08	Hword	PID	This field holds the PID of the subframe associated with this BD.

Table 37-32. Sub-Frame Buffer Descriptor Field Descriptions

Offset	Bits	Field	Description
0x0A	Hword	Index (to DFT)	This field holds the index to the DeMUX fragment table (DFT). This direct the host to the fragment on which this sub frame has been received. Once all the sub frames are processed the host has to change the STATE of this entry. The address to the entry is calculated in the following way: External DFT Base pointer+index*32.
0x0C	Word	Time stamp	This field holds the time stamp of the fragment / PPP mux frame.

37.19.5 Mux Encapsulation Management Table (MMT)

This table contains a list of the active queues in the system that are configured to work in PPP mux mode and require building superframes.

Note that there can't be more than 16 Mux queues in the system.

The size of the table is determined by the number of PPP mux active queues. The bigger the size of the table, the higher the latency of building the superframe for each queue.

Figure 37-45 describes the management table for the mux process. It is pointed to from the MCC global parameter RAM (Table 37-5). The table size is 56 bytes.

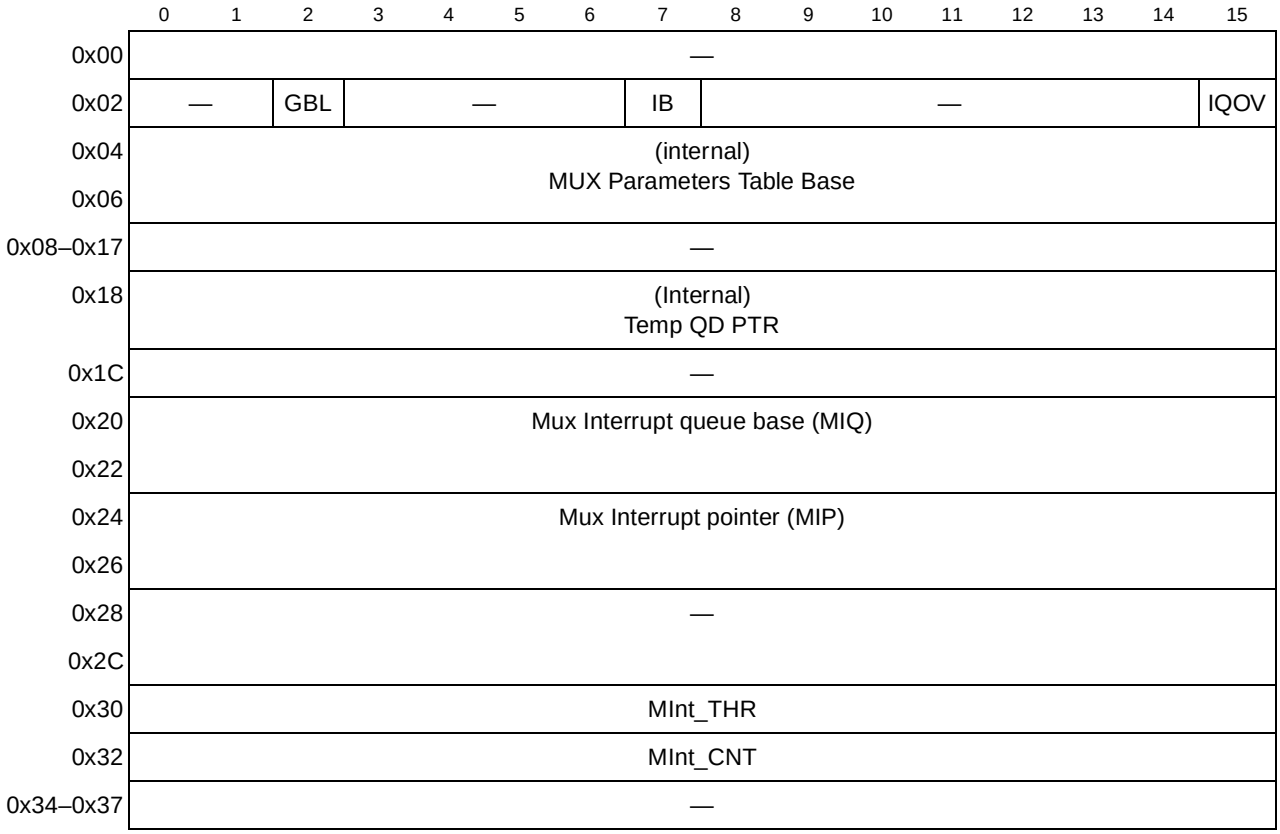


Figure 37-45. Mux Encapsulation Management Table

Table 37-33. Mux Encapsulation Management Table Field Descriptions

Offset	Bit	Size	Name	Description
0x00		Hword	—	Reserved. Initialize to zero.
0x02	0–1	Hword	—	Reserved. Should be cleared.
	2		GBL	Global. Setting GBL enables snooping of Interrupt queues. This should match BMR[GBL]. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "Serial DMA Mode Register (SDMR)."
	3–6		—	Reserved. Should be cleared.
	7		IB	IB- Interrupt Bus Selection for the Mux-encapsulation process. 0 Resides on the coherent system bus. 1 Resides on the secondary bus.
	8–14		—	Reserved. Should be cleared.
	15		IQOV	IQOV Indicates the event of an Interrupt queue Overflow. When there is a Mux Encapsulation event set in the CEVTER register for the Mux process (Based on its selected SNUM), the application should read this status bit which indicates a situation where there is no valid entry in the interrupt queue. The host should never reset this bit. It will be cleared by the QUICC Engine block when the full condition of the interrupt queue is gone.
0x04		Word	MPT Base (Internal)	MUX Parameter Table Base. Base address of the MUX encapsulation parameter table described in 37.19.6. Should be 64 bytes aligned.
0x08		16 Bytes	—	Reserved. Initialize to zero.
0x18		Word	Temp QD PTR (Internal)	A pointer to internal memory where the external Queue descriptor is fetched when using external QD. The user should initialize this field with a pointer to 16 bytes in the Multi-user RAM.
0x1C		Word	—	Reserved. Initialize to zero.
0x20		Word	Mux Interrupt queue base pointer (MIQ)	This field holds the pointer to the interrupt queue base. The interrupt queue is dedicated to the mux interrupts. Pointer to external memory.
0x24		Word	Mux Interrupt Queue pointer (MIP)	Pointer to the Mux Interrupt queue. The RISC writes the next interrupt information to this entry when an exception occurs. Initialize to Mux Interrupt queue base pointer. Pointer to external memory.
0x28–0x2F		8 Bytes	—	Reserved. Initialize to zero.
0x30		Hword	MInt_THR	Mux fragment receive Interrupt threshold.
0x32		Hword	MInt_CNT	Mux fragment receive count. Internal use. Should be initialize to MInt_THR
0x34		Word	—	Reserved. Initialize to zero.

37.19.6 Mux Encapsulation Parameters Table

This table resides in the internal multi-user RAM. Each transmit queue (QD) which is designated to work in Mux mode is associated with a PPP Mux Parameters Table, shown in Figure 37-46. Each entry in the

table contains all the information needed for the mux process to generate a PPP mux super-frame.It allocates space for parameters which are initialized by the host and space for the internal state and variables used by the RISC when processing a superframe. The size of the table is determined by the number of PPP mux QDs—one entry per QD. A Wrap bit indicates the last entry in the table. Each entry in the table is 64 bytes long. The Mux encapsulation process will go through the table entries and for each entry, it builds a Mux super-frame.

0x00		Mux Encapsulation Params #1
0x40		Mux Encapsulation Params #2
.....		.....
0xXX	W	Mux Encapsulation Params #N

Figure 37-46. Mux Parameters Table (MPT)

The process scans through the entries in a round robin fashion. When reaching the entry marked with the W indication, the RISC wraps to the first entry for processing. As long as a superframe is not valid the queue which is related to the request entry will be marked as empty. Once it is ready, the QE (Queue Empty) bit is negated and the queue could be selected by the WFQ algorithm.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	MMR															
0x02	Def_PID															
0x04	Super_Frame_Size															
0x06	—															
0x08	TimeOUT_Period															
0x0A																
0x0C	Age Period															
0x0E																
0x10	Tx_SuperF_Discard_CNT															
0x12																
0x14	SuF_Discard_CNT															
0x16																
0x18	—	UDH En	UDH_Index								UDH_Size					
0x1A	Mux Encapsulation Offset out															
0x1C	QD_PTR															
0x1E																
0x20	Mux_req	—			—			—			—	Link	Link ID/Bundle ID			W
0x22	Cur_PID															
0x24	Super Frame Timestamp															
0x26																
0x28	FBPT_PTR (Internal)															
0x2A																
0x2C	—															
0x2E																
0x30	—															
0x32	—															
0x34	State															
0x36																
0x38 - 0x3F	—															

Table 37-34. MPT Entry

Table 37-35. PPP MUX Parameter Table (MPT) Entry Descriptions

Offset	Bits	Name	Description
0x00	Hword	MMR	Mux Mode Register. See Table 37-36.
0x02	Hword	Def_PID	Subframe default PID. The default value for the subframe PID. If the PID of the first subframe selected is equal to this value, then PFF of the subframe header would be 0, and no PID would be asserted.
0x04	Hword	Super_Frame_Size	MUX super frame size. Defines the maximum transmitted size of a PPP MUX superframe. The size specified in this field does not include the PPP MUX PID which could be of one or two bytes depending on the mode in which the bundle is programmed. In case UDH mode is enabled, the size of the UDH is read from the QD and is subtracted from the value of MUX_Super_Frame_Size. Programming a large value (>MRU) enables encapsulation of multi fragment superframe.
0x06	Hword	—	Reserved, initialized to 0.
0x08	Word	TimeOUT_Period	MUX Time-out Period. The Time-out period for the RISC for stopping the multiplexing process of a superframe. This value is valid when TimeOutEn in the Mux Mode register is on. The user should initialize this field as an integer value of time units which are derived from CETSCR programming, see Section 20.3.9, “QUICC Engine Time-Stamp Control Register (CETSCR).”
0x0C	Word	Age Period	MUX aging Period. This value determines the maximum time allowed for a PPP MUX subframe to be in a queue, before it is discarded. If the Age_En mode is on, this value represents the amount of time allowed. See Table 37-29 in MUX Time Stamp field, for more information. The user should initialize this field as an integer value of time units which are derived from CETSCR programming, see Section 20.3.9, “QUICC Engine Time-Stamp Control Register (CETSCR).”
0x10	Word	Tx_SuperF_Discard_CNT	Transmitter Mux Super Frame Discard counter. Initialized to 0. Counts the number of Super frame that are discarded due to the Link Aging mechanism.
0x14	Word	SuF_Discard_CNT	MUX Encapsulation SubFrame Discard counter. Initialized to 0. Counts the number of SubFrames that were discarded due to the Aging expiration by the MUX process.
0x18	0	—	Reserved. Should be cleared.
	1	UDHEn	User Defined Header (UDH) Enable. 0- No User Defined Header is appended to the MUX fragment/frame. 1- User Defined Header is appended to the MUX fragment/frame. The UDH is appended before the PPP MUX PID (0x0059).
	2–7	UDH_index	Index of UDH. Index from UDH Base. The UDH Base is taken from the BPT, see Table 37-9. Address to the UDH is (UDH_Index*0x100+UDH Base). There are up to 64 different UDH under each bundle.
	8–15	UDH_Size	Size-1 in bytes of the UDH (UDH=255 means UDH of 256 bytes and UDH=0 means UDH of 1 byte). Note that UDH_Size has to be smaller than MFS parameter in case of transmitting ML MC PPP fragments, see field in BPT.
0x1A	Hword	Mux Encapsulation Offset out	Mux Encapsulation Offset out. The Offset in the queue BD ring of the MUX process. Should be initialized to 0.

Table 37-35. PPP MUX Parameter Table (MPT) Entry Descriptions (continued)

Offset	Bits	Name	Description
0x1C	Word	QD_PTR (external or internal)	Queue Descriptor Pointer. This is the pointer to the queue which is requesting the MUX encapsulation services. The queue could be in the internal or external memory and the bus location is determined in the Mux Encapsulation Parameter Table [MMR].
0x20	0	Mux_req	Mux request. Must be initialize to 1. RISC internal use.
	1–10	—	Reserved. Should be cleared.
	11	Link	0- This entry is associated with a ML queue in a bundle specified in bits 12-14. 1 -This entry is associated with a link which is link ID is specified in bits 12-14. This information is initialized by the host.
	12–14	Link ID/ Bundle ID	Specify the Link ID /Bundle ID for this entry. Should be initialized by the host. This information is copied to the interrupt queue upon a MUX event.
	15	W	Wrap (last request in Mux encapsulation request table) 0 This is not the last request in the table. 1 This is the last request in the Mux encapsulation request table.
0x22	Hword	Cur_PID	Current Subframe PID value. Initialized to the MUX_Def_PID parameter. Used by the RISC to determine the PFF value for the next subframe.
0x24	Word	Super Frame Timestamp	Super Frame Timestamp. User initialized to 0. Used by the Mux process to indicate the time stamp of the sub-frame which has the highest timestamp value (enqueued the latest) in the whole superframe. If Aging is enabled and the aging period has elapsed, the complete superframe will be discarded when processed by the transmitter back-end (or the link). This entry which is the MUX process output, is used as input to the transmitter back-end (or the link) and should be used/save by the transmitter back-end (or the link) before raising Mux_req.
0x28	Word	FBPT_PTR (Internal)	Points to the Free Buffer Pool Parameter Tables that this MUX queue is using. Valid only if FBP mode is enabled in the MUX TQD. Four different FBP are supported on the transmitter. The address specified in this entry should match the address calculated by: BPT[TXFBPT_BASE] + TQD[Tx_FBP_Sel]. The MUX encapsulation will use this entry when returning a buffer pointer after discarding a sub-frame due to aging.
0x2C–0x33	8 Bytes	—	Reserved, should be cleared.
0x34	Word	State	RISC internal state for this request entry. Should be initialize to 0x80000000
0x38	8 Bytes	—	Reserved, should be cleared.

37.19.7 Mux Mode Register (MMR)

The Mux mode register (MMR) is a user-initialized register, shown in Figure 37-47.

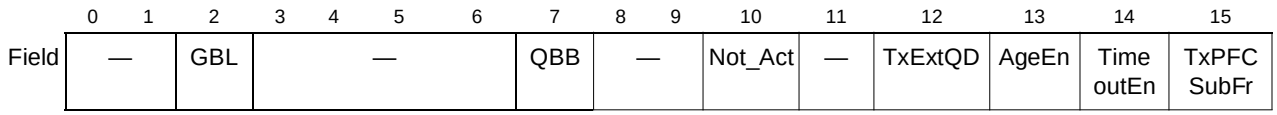


Figure 37-47. Mux Mode Register (MMR)

MMR fields are described in Table 37-36.

Table 37-36. MMR Field Descriptions

Bits	Name	Description
0–1	—	Reserved. Should be cleared.
2	GBL	Global. Setting GBL enables snooping of BD's and QD's. This should match BMR[GBL]. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "Serial DMA Mode Register (SDMR)."
3–6	—	Reserved- Should be cleared
7	QBB	Queue Descriptors and Buffer Descriptors bus Selection 0 Reside on the coherent system bus. 1 Reside on the secondary bus. This should match the bus selection of the QD given in BMR[BDB].
8–9	—	Reserved. Should be cleared.
10	Not_Act	Not Active- This bit signals to the QE that this MUX queue is currently not active and there is no need to process a superframe for this queue. The host should clear this bit when the queue is operational.
11	—	Reserved. Should be cleared.
12	TxExtQD	Transmitter External Queue Descriptor. 0- Indicating the MUX process that the queue is in internal memory 1- The queue resides in external memory.
13	AgeEn	Age Enabled. Valid only for PPP MUX superframes. Upon completion the encapsulation of a superframe, before transmission, the current time is compared to timestamp of the "youngest" subframe in the superframe. If the time elapsed is bigger then the Age Period the complete superframe is discarded. 0 - The subframes age algorithm is disabled. 1 - The subframes age algorithm is enabled.
14	TimeoutEn	Timeout Enable. During encapsulation of PPP MUX subframe the QE examines the time stamp value placed by the host on each BD. If the time difference between current time and the BD timestamp value exceeds the Age Period value, the subframe is discarded (and marked as such in the BD) 0 - The time-out MUX mechanism is disabled. 1 - The time-out MUX mechanism is enabled.
15	TxPFCSubFr	Transmit PID Compressed for the PPP MUX sub frame header. 0 - PID transmitted is 2 bytes length. 1 - PID transmitted is 1 byte length (only if MSB byte of the PID is 0).

37.20 PPP Exceptions

The MCC interrupt handling involves two data structures: the MCCE (MCC event register) and the circular interrupt queues.

Fourteen priority interrupt queues are available. The queues are available for legacy MCC usage using the MCC programming model or, alternatively, are available for the PPP application. In this case the programming model is different. It is the application responsibility to use each queue for a unique purpose since the behavior is completely different in each of the cases. If, for example interrupt queue #1 is used for legacy MCC operation it should not be configured in any of the Link or Class parameters LMR/CMR[RIQN] as a queue for PPP usage.

The host controls the number of interrupts sent to the core using a 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 this counter reaches zero, the global interrupt MCCE[INTx], is set.

37.20.1 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in the Interrupt Queue Parameter Tables.

When the microcode generates an interrupt request, the RISC writes a new entry to the interrupt queue, the valid (V) bit is set and the queue pointer (INTQ_PTR) increments. When INTQ_PTR reaches a location with the W bit set, it wraps to the first entry in the queue.If the RISC tries to overwrite a valid entry (V=1), an overflow condition occurs and the queue's overflow flag of the corresponding queue, MCCE[QOVx], is set.

37.20.2 Interrupt Queue Entry

Each interrupt queue entry is 8 bytes long. Each interrupt queue entry provides detailed interrupt information to the host.

An interrupt queue entry is shown in Figure 37-48.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	V	W	MRU	Rx BP_BSY	NIDL	IDL	BP_RLI	FLOSS_TH	PRT_BSY	DM_BSY	UN	FLOSS	WBD_BSY	BSY	TXB	RXF
0x02	Class id			—			Tx BP_BSY	Tx_FBP_Sel		LCP	Bundle ID			Link ID		
0x04	Queue Descriptor pointer/Tx BP Busy Buffer Pointer															
0x06																

Figure 37-48. Interrupt Queue Entry

Table 37-37 describes the interrupt queue entry fields.

Table 37-37. PPP Interrupt Queue Entry Fields Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	W	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	2	MRU	A fragment/frame was received with MRU violation.The Link ID field specified on which Link it happened. Mask for this interrupt is defined in LINTMSK register in the LPT

Table 37-37. PPP Interrupt Queue Entry Fields Descriptions (continued)

Offset	Bits	Name	Description
	3	Rx_BP_BSY	Buffer Pool Busy. Set when a free buffer pool Busy condition occurs. It means no buffer is available for reception. Important: This is a special event and will generate an interrupt regardless of the condition of the counter reaching the desired threshold.
	4	NIDL	Non idle state. The line is not idle.
	5	IDL	Idle state. This line is idle.
	6	BP_RLI	Buffer Pool RLI. Set when a free buffer pool has reached the level of the RLI threshold. Important: This is a special event and will generate an interrupt regardless of the condition of the counter reaching the desired threshold.
	7	FLOSS_TH	Fragment loss due to threshold reached. See Figure 37-25. A fragment loss was detected by the receiver. The packet was discarded. The Bundle ID field and the Class ID field provide to the user information about the WBD ring it happened to. This interrupt can not be masked.
	8	PRT_BSY	Packet Reconstruction Table Busy condition. The Packet Reconstruction table is busy. Packets were discarded due to this condition. This interrupt can not be masked.
	9	DM_BSY	De-mux Busy condition. The Demux table is busy. Fragments/frames were discarded due to this condition. This interrupt can not be masked. This implies that during the Demux process some sub-frames are lost.
	10	UN	Tx no data. The RISC sets this flag if there is no data available to be sent to the transmitter. The transmitter sends an ABORT indication and then sends idles. The Link ID field indicates on which link it happened.
	11	FLOSS	Fragment loss occurance.There was a fragment loss detected by the receiver.Packet was discarded.The Bundle ID field and the Class ID field provide to the user information about the WBD ring it happened to.This interrupt can not be masked.
	12	WBD_BSY	Busy condition in WBD ring.The WBD ring associated to the class specified in the Class ID field, is busy.Packets were discarded due to this condition. This interrupt can not be masked.
	13	BSY	Busy condition. The RxBD table, associated with the queue specified in the Queue Descriptor field, is busy. Packets were discarded due to this condition. This is an unmaskable interrupt
	14	TXB	TX buffer. A buffer has been prepared for transmission and the corresponding BD is free to get a new Data buffer from the host.
	15	RXF	RX frame. A frame was received in the associated queue. This interrupt can be mask by clearing the INTMSK bit in the Queue Descriptor.
0x02	0-2	ClassID	Class ID. Indicates the Class where events like RxF, TxB, WBD_BSY, FLOSS, PRT_BSY and DM_BSY occurred.
	3–5	—	Reserved
	6	Tx_BP_BSY	Tx Buffer Pool Busy. Set when a Transmitter free buffer pool Busy condition occurs. It means no room is available for the buffer return on transmission. Important: This is a special event and will generate an interrupt regardless of the condition of the counter reaching the desired threshold.
	7–8	Tx_FBP_Sel	Tx_FBP_Sel- Indicates the Buffer Pool on which the busy conditions occurred.
	9	LCP	Indicates LCP frame reception, MRU violation of LCP frame or LCP queue Busy.
	10–12	Bundle ID	Bundle ID. Indicates the Bundle where events like RxF, TxB, WBD_BSY, PRT_BSY, FLOSS, BP_BSY, BP_RLI and DM_BSY occurred.
	13–15	Link ID	Indicates the LinkId where events like TXBD (if occurred on Link), MRU,DM_BSY (if occurred on Link), UN, RXF (if occurred on Link). In a ML/MC event, this field is not valid.

Table 37-37. PPP Interrupt Queue Entry Fields Descriptions (continued)

Offset	Bits	Name	Description
0x04	Word	Queue Descriptor/ Tx BP Busy Buffer Pointer	This field contains the pointer to the Queue Descriptor where the event occurred. In case of Transmit Buffer Pool busy, this entry contains the buffer pointer which has no room in the FBP due to FBP_Busy conditions.

37.20.3 Interrupt Queue Parameter Tables

The interrupt queue parameters are held in parameter tables in the multi-user RAM; PPP_INTBASE in the Global MCC parameter points to the base address of these tables. Each of fourteen interrupt queues has its own parameter table with starting address given by PPP_INTBASE+IntQ_num*32.

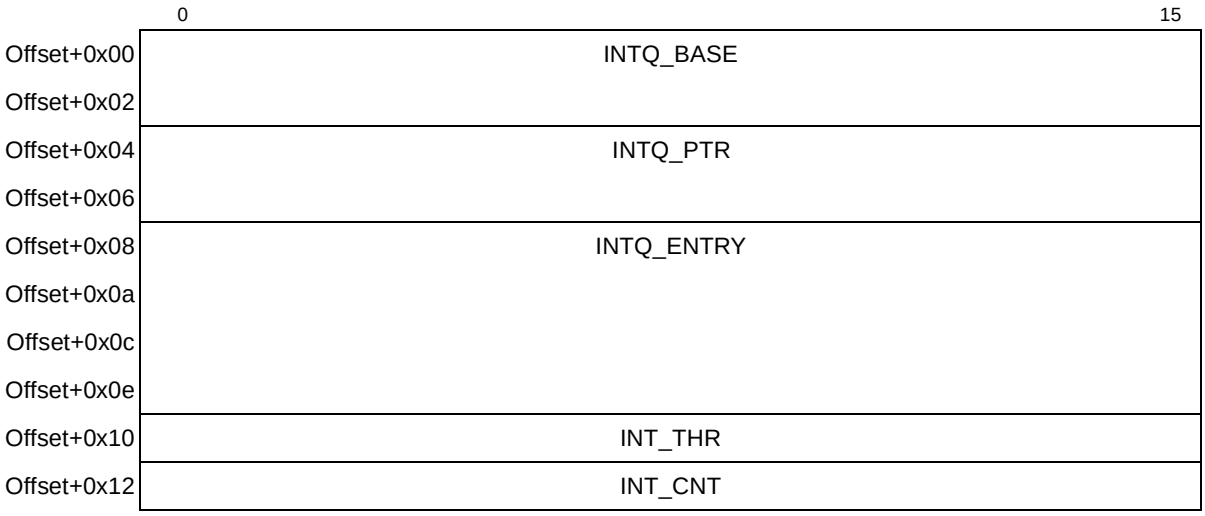


Figure 37-49. Interrupt Queue Parameter Table

Each fields of this table is described in Table 37-38.

Table 37-38. Interrupt Queue Parameter Table Field Descriptions

Offset	Name	Width	Description
0x00	INTQ_BASE	Word	Base address of the interrupt queue. User-defined.
0x04	INTQ_PTR	Word	Pointer to interrupt queue entry. Initialize to INTQ_BASE.
0x08	INTQ_ENTRY	8 bytes	Interrupt queue entry. Must be zero.
0x10	INT_THR	Hword	Interrupt threshold. User-defined global interrupt threshold: the number of interrupts required before the RISC issues a global interrupt (MCCE[INTx]).
0x12	INT_CNT	Hword	Interrupt counter.Initialize with INT_THR, The RISC decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag MCCE[INTx] is set (and MCCE[8] also).

37.20.4 MCC Event and Mask Registers

Since legacy MCC operation concurrent to the PPP operation is enabled there are two sets of interrupts and interrupts handling. The legacy MCCE is defined to support up to 7 interrupt queues. Each queue could be managed by a different interrupt management since the interrupt entries are different on legacy MCC operation and on PPP operation. It is the user responsibility to use an interrupt queue in a single mode. Trying to use a single queue for MCC and PPP causes erroneous results.

In legacy MCC code, four queues are available for receive operation (queue0-queue 3) and a single queue is available for transmit events per MCC. Processing the interrupts and interrupt queues is described in the MCC global parameter table, see Section 34.2.1, “Global MCC Parameters.”

In PPP mode, numbering the queues still applies but the management is located in the multi-user RAM and pointed to by PPP_INTBASE described in Figure 37-20. An event which is related to a **link** (like MRU violation or reception of plain PPP frame) generates an interrupt if not masked on the MCCE register. The selection of which queue to use is defined in the LMR[RIQN] for the receiver. If there is no legacy MCC code running at the same application, the queues available are 0-3, 6-10 and 13 for the receiver.The transmitter is limited and always uses queue #4 or #11 for link events according to LMR[TI_Low] bit.

An event which is related to **multi-link multi-class** traffic will act the same. The receiver can select a queue using the CMR[RIQN]. If there is no legacy MCC code running at the same application, the queues available are 0-3, 6-10 and 13 for the receiver. The transmitter is limited to using queue #5 or # 12 for multilink multi class events according to CMR[TI_Low] bit.

It is forbidden to mix queues of **link** events and **multi-link multi-class** events that are generated from background tasks. This implies that for multilink multiclass operation there should be for Link events at least one queue for receive, and one for transmit. This is true also for multilink events (background events). This gives a minimum of four queues for a multilink multiclass operation.

The MCCE/MCCM register is described below. For each bit the corresponding bit in the MCCM register enables the interrupt.

Offset:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	QOV0	GINT 0	QOV1	GINT 1	QOV2	GINT 2	QOV3	GINT3	QOV4	GINT4	QOV5	GINT5	GQOV6	GINT6	GUN	GOV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	QOV7	GINT 7	QOV8	GINT 8	QOV9	GINT 9	QOV10	GINT10	QOV11	GINT11	QOV12	GINT1 2	QOV13	GINT13	CUN	Res
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 37-50. MCC Event Register (MCCE)/Mask Register (MCCM)

Table 37-39 describes MCCE fields.

Table 37-39. MCCE/MCCM Register Field Descriptions

Bits	Name	Description
0	QOV0	PPP interrupt queue x overflow (QOVx). Set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number x. Global Interrupt (GINTx). Set when the number of interrupt issued in the interrupt queue x reaches the pre-defined threshold. These four queues are available for receive events on standard MCC operation and for PPP operation for link and class events (for PPP is selected based on CPT[RIQN] or LPT[RIQN]).
1	GINT0	
2	QOV1	
3	GINT1	
4	QOV2	
5	GINT2	
6	QOV3	
7	GINT3	
8	QOV4	PPP interrupt queue 4 overflow. QOV4 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 4.
9	GINT4	Global Interrupt. Set when the number of interrupt issued in the interrupt queue 4 reaches the pre-defined threshold. This queue is dedicated for transmitter Link events when LPT[TI_Low] is cleared.
10	QOV5	PPP interrupt queue 5 overflow. QOV5 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 5.
11	GINT5	Global Interrupt. Set when the number of interrupt issued in the interrupt queue 5 reaches the pre-defined threshold. This queue is dedicated for transmitter Class events when CPT[TI_Low] is cleared.
12	QOV6	PPP interrupt queue 6 overflow. QOV6 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 6.
13	GINT6	Global Interrupt. Set when the number of interrupt issued in the interrupt queue 6 reaches the pre-defined threshold. This queue is dedicated for legacy MCC code transmitter event or for receiver PPP link or class event (for PPP is selected based on CPT[RIQN] or LPT[RIQN]).
14	GUN	Global transmit underrun. This flag indicates whether an underrun occurred inside the MCC's transmit FIFO array. The user must clear this bit.
15	GOV	Global receiver overrun. This flag indicates whether an overrun occurred inside the MCC's receive FIFO array. The user must clear this bit.
16	QOV7	**Note:** Queue 7 OverFlow. Set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table.
17	GINT7	This queue is available for PPP operation and is considered as an extra. This queue is available for PPP operation for link and class events and is selected based on CPT[RIQN] or LPT[RIQN]. **Note:** Global Interrupt (GINTx). Set when the number of interrupt issued in the interrupt queue x reaches the pre-defined threshold.
18	QOV8	Queue 8 OverFlow. Set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table.

Table 37-39. MCCE/MCCM Register Field Descriptions (continued)

Bits	Name	Description
19	GINT8	This queue is available for PPP operation and is considered as an extra. This queue is available for PPP operation for link and class events and is selected based on CPT[RIQN] or LPT[RIQN]. Global Interrupt (GINTx). Set when the number of interrupt issued in the interrupt queue x reaches the pre-defined threshold.
20	QOV9	Queue 9 OverFlow. Set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table.
21	GINT9	This queue is available for PPP operation and is considered as an extra. This queue is available for PPP operation for link and class events and is selected based on CPT[RIQN] or LPT[RIQN]. Global Interrupt (GINTx). Set when the number of interrupt issued in the interrupt queue x reaches the pre-defined threshold.
22	QOV10	Queue 10 OverFlow. Set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table.
23	GINT10	This queue is available for PPP operation and is considered as an extra. This queue is available for PPP operation for link and class events and is selected based on CPT[RIQN] or LPT[RIQN]. Global Interrupt (GINTx). Set when the number of interrupt issued in the interrupt queue x reaches the pre-defined threshold.
24	QOV11	PPP interrupt queue 11 overflow. QOV11 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 11.
25	GINT11	Global Interrupt. Set when the number of interrupt issued in the interrupt queue 11 reaches the pre-defined threshold. This queue is dedicated for transmitter Link events when LPT[TI_Low] is set.
26	QOV12	PPP interrupt queue 12 overflow. QOV12 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 12.
27	GINT12	Global Interrupt. Set when the number of interrupt issued in the interrupt queue 12 reaches the pre-defined threshold. This queue is dedicated for transmitter Class events when CPT[TI_Low] is set.
28	QOV13	PPP interrupt queue 13 overflow. QOV13 is set (and an interrupt request generated) by the RISC whenever an overflow occurs in the PPP circular interrupt table number 13.
29	GINT13	This queue is available for PPP operation and is considered as an extra. This queue is available for PPP operation for link and class events and is selected based on CPT[RIQN] or LPT[RIQN]. Global Interrupt. Set when the number of interrupt issued in the interrupt queue 13 reaches the pre-defined threshold.
30	CUN	Channel Transmit Underrun.
31	Res	Reserved

37.20.5 PPP Demux Interrupt Queue Entry

The de-mux process has it's own event register and queue management. The management of the interrupt queue is done in the Demux Management Table (DMT) described at Table 37-30. Masking the interrupts is defined in the queue descriptor of the demux queue by the INTMSK bit. See Figure 37-31.

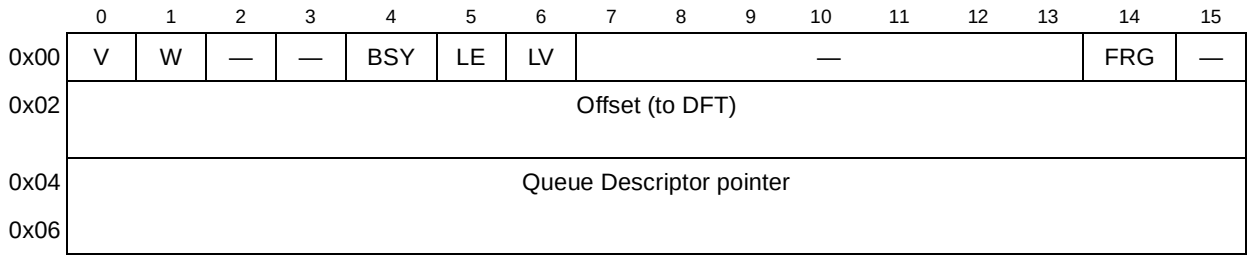


Figure 37-51. Demux Interrupt Queue Entry

Table 37-40 describes PPP DeMux Interrupt Queue Entry fields.

Table 37-40. PPP DeMux Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	W	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	2–3	—	Reserved. Should be cleared.
	4	BSY	Busy condition. The RxBD table associated with this sub-frame is busy. sub-frames were discarded due to this condition.
	5	LE	Length Error. Indicates a Length Error condition on a sub-frame. It happens when a sub-frame length exceeds the length of the whole super-frame.
	6–13	—	Reserved. Should be cleared.
	14	FRG	For Termination mode: Rx PPP mux fragment interrupt. A fragment was received and one or more sub-frame were extracted by the de-mux process. In case of an error two bits are set. .
	15	—	Reserved. Should be cleared.
0x02	Hword	Offset (to DFT)	This field holds the offset to the DeMUX fragment table (DFT) from the DFT base. This directs the host to the fragment on which this sub frame has been received. Once all the sub frames are processed the host has to change the STATE of this DFT entry. The address to the entry is calculated in the following way: DFT pointer+(offset*0x20).
0x04	Word	QD pointer	This field holds the pointer to the queue Descriptor to which the Sub Frames were assigned to.

37.20.6 PPP Mux Interrupt Queue Entry

The mux process share the event register with the demux. The management of the interrupt queue is done in the Mux Management Table described at Table 37-30. Masking the interrupts is defined in the queue descriptor of the mux queue by the INTMSK bit. See Figure 37-32. Figure 37-52 illustrates the interrupt queue entry for mux process.

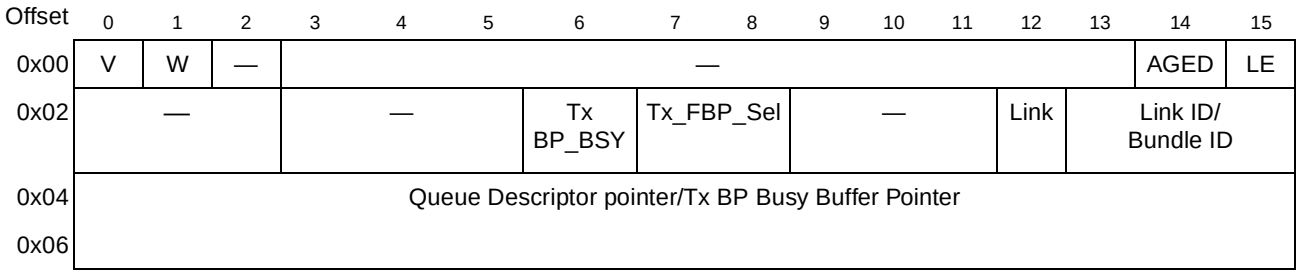


Figure 37-52. Mux Interrupt Queue Entry

Table 37-41 describes PPP Demux Interrupt Queue Entry fields.

Table 37-41. PPP Mux Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	2	W	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	2–13	—	Reserved. Should be cleared.
	14	AGED	A Tx BD was dropped due to aging mechanism if enabled.
	15	LE	Length error indication. The length of the sub frame exceeds the max allowed length of a sub-frame.
	0–5	—	Reserved.
	6	Tx_BP_BSY	Tx Buffer Pool Busy. Set when a Transmitter free buffer pool Busy condition occurs. It means no room is available for the buffer return on transmission. Important: This is a special event and will generate an interrupt regardless of the condition of the counter reaching the desired threshold.
	7–8	Tx_FBP_Sel	Tx_FBP_Sel - Indicates the Buffer Pool on which the busy conditions occurred.
	9–11	—	Reserved.
	12	Link	0-Indicates a ML event. 1-Indicates a link event.
	13–15	Link/ Bundle ID	The link ID on which the event occurred in case this is a link event, or the Bundle ID in ML/MC setup.
0x04	Word	Queue Descriptor/ Tx BP Busy Buffer Pointer	This field contains the pointer to the Queue Descriptor where the sub-frame event occurred. In case of Transmit Buffer Pool busy this entry contains the buffer pointer which has no room in the FBP due to FBP_Busy conditions.

37.20.7 PPP Mux/Demux Event Register

The Mux and Demux processes share the CEVTER event register see Section 20.4.4, “RISC Timer Event Register (CETER)/Mask Register (CETMR).” This register contains the event bit which reports events recognized by the Mux and the Demux tasks. On recognition of an event the task sets its corresponding VTxx bit. The bit is determined by the SNUM of the process which defines it’s VT (Virtual Task) number.

In the event of an Interrupt queue overflow there is an indication in the MMT and in the DMT tables respectively. The bit IQOV is set and indicates the event that there is no space left in the interrupt queue for a new entry. Theses bits are status bits from the QUICC Engine block and should never be cleared by the host. They will be updated by the QUICC Engine block upon every event.

37.21 Bus Selection

Table 37-42 describes the location of the bits which control the bus selection of each external data structure.

Table 37-42. Bus Selectors Location

Structure	Location
Data buffers	BMR[DTB]: Bundle Mode Register, bit 6.
BD’s	BMR[BDB]: Bundle Mode Register, bit 7
WBD ring	BMR[BDB]: Bundle Mode Register, bit 7
TX Free Buffer Pool tables	BMR[BDB]: Bundle Mode Register, bit 7
RX Free Buffer Pool table	BMR[BDB]: Bundle Mode Register, bit 7
Interrupt queues	BMR[BDB]: Bundle Mode Register, bit 7
Queue Descriptors	BMR[BDB]: Bundle Mode Register, bit 7
Fragment Descriptors	BMR[BDB]: Bundle Mode Register, bit 7
Demux Fragment table	DMX_MNG[BIB]: Demux Management Table, offset 0, bit 7
Demux Interrupt Queue	DMX_MNG[BIB]: Demux Management Table, offset 0, bit 7
Mux Interrupt Queue	Mux Management Table, offset 0x02, bit 7 [IB]

37.22 MCC Initialization and Commands for PPP Mode

The following is a general sequence for initializing the MCC and its channels after reset.

- Program the parallel I/O port interface for the TDMs to be used.
- Program the SI’s SIRAM and related registers.
- Initialize all MCC global parameters.
- Initialize MCC channels specific parameters corresponding to each Link Parameter Table (LPT) used. Also configure all relevant data structures.
- Initialize the Super-channel Table.
- Enable Super-channel mode for the MCC transmitter.

- Issue MCC INIT TX according to the number of channels in the setup.
- Issue MCC INIT RX according to the number of channels in the setup.
  — (Could use MCC INIT RX AND TX command as well instead of the two last steps)
- Issue INIT PPP Rx & Tx Link commands for each link (described below.)
- Issue INIT PPP Rx/Tx BACK-END commands as described below if Rx/Tx ML MC or De-MUX processes are enabled.
- If required, issue INIT PPP MUX/De-MUX Processes command as described below.
- Enable the TDM.

37.22.1 PPP Host Commands

When working in ML/MC PPP mode, the following commands must be issued by writing to CECR. This command enables the three tasks working as back-end in the PPP microcode. The commands should be issued before enabling any PPP operation.

All MCC commands are still valid and the new commands should be issued in addition to the PPP specific commands. The reason for this is the fact that the old MCC commands are resetting the MCC channels' FIFOs while the new commands do not effect the FIFO.

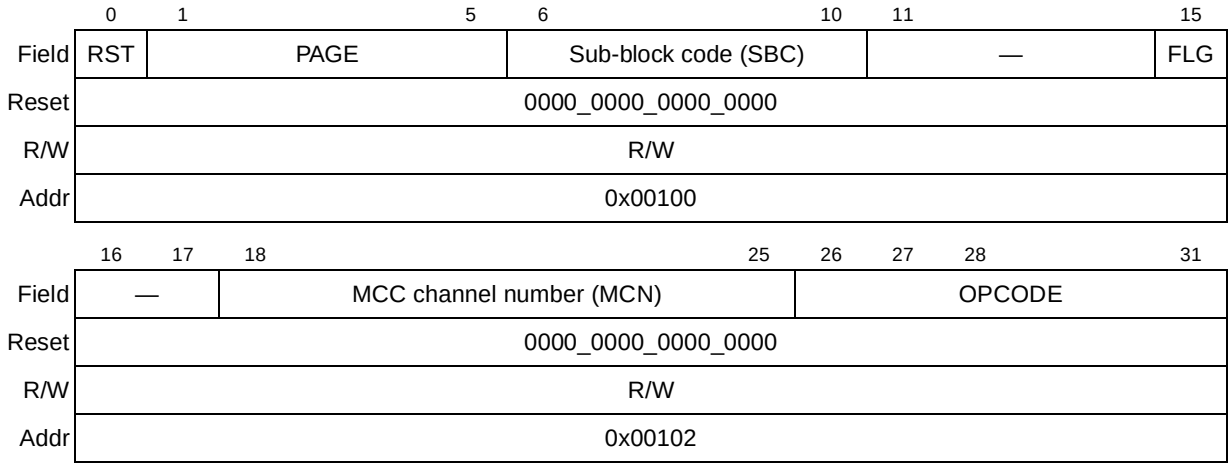


Figure 37-53. QUICC Engine Command Register (CECR)

Table 37-43. CECR Field Descriptions

Bit	Name	Description
0	RST	Software reset command. Set by the core and cleared by the RISC. When this command is executed, RST and FLG bit are cleared within two general system clocks. The QUICC Engine reset routine is approximately 60 clocks long, but the user can begin initialization of the QUICC Engine block immediately after this command is issued. RST is useful when the core wants to reset the registers and parameters for all the channels (UCCs, SPI, MCC) as well as the RISC and RISC timer tables. However, this command does not affect the serial interface (SI) or parallel I/O registers.
1–5	PAGE	Indicates the parameter RAM page number associated with the sub-block being served. See the SBC description for page numbers.

Table 37-43. CECR Field Descriptions (continued)

Bit	Name	Description
6–10	SBC	Sub-block code. Set by the core to specify the sub-block on which the command is to operate. For a listing of the SBC values, see Table 20-3. Set according to OPCODE[28-31]. Refer to Table 37-44.
15	FLG	Command semaphore flag. Set by the core and cleared by the RISC. 0 The RISC is ready to receive a new command. 1 The CECR contains a command that the RISC is currently processing. The RISC clears this bit at the end of command execution or after reset.
16–17	—	Reserved
18–25	MCN	MCC channel number. Specifies the first channel number associated to the link in case of MCC-PPP command.
26–31	OPCODE	Operation code. This field is extended from 4 bits to 6 bits for the PPP protocol. For the PPP operation the following opcodes are valid: INIT PPP Rx link - Opcode= 0b'011011 INIT PPP Tx link - Opcode= 0b'001011 INIT PPP Rx & Tx Link - Opcode= 0b'101011 INIT PPP Rx back-end - Opcode= 0b'011101 INIT PPP Tx back-end - Opcode= 0b'001101 INIT PPP back-end - Opcode= 0b'101101 INIT DE-MUX Process- Opcode= 0b'011110 INIT MUX Encapsulation Process- Opcode=0b'001110 INIT PPP MUX Processes - Opcode=0b'101110

The following table describes the PPP host commands:

Table 37-44. PPP Commands

Command	Description
INIT PPP Rx link	This command initializes the correct state for PPP receiver operation. MCN should be point to the channel which is mapped to the LPT. The host should issue this command for each link in the system. It should be issued AFTER the MCC INIT Rx command.
INIT PPP Tx link	This command initializes the correct state for PPP transmitter operation. MCN should be point to the channel which is mapped to the LPT. The host should issue this command for each link in the system. It should be issued AFTER the MCC INIT Tx command.
INIT PPP Rx & Tx Link	This command combines both previous commands.
INIT PPP Rx back-end	Initializes the Rx back-end tasks of the PPP. This commands initializes the Rx Packet reconstruction task. Should be issued if Rx ML MC is enabled or De-MUX operation is needed.
INIT PPP Tx back-end	Initializes the Tx back-end tasks of the PPP. This commands initializes the Tx fragmentation process. Should be issued if Tx ML MC is enabled.
INIT PPP back-end	Initializes the Tx back-end and the Rx back-end tasks. Combines the both previous commands
INIT DE-MUX Process	Initializes the Rx De-Mux process This command should be issued following the back-end initialization
INIT MUX Encapsulation Process	Initializes the Tx MUX encapsulation process.
INIT PPP MUX Processes	Initializes both, the Tx MUX encapsulation process and the receive de-mux process.

37.22.2 Dynamic Addition and Removal of a Link

The user may want to dynamically add links into a specific bundle, or remove links from a bundle. The user should allocate memory space which can accommodate the maximum number of links expected to be active in the bundle. When initializing the data structures a smaller number can be used, but it enables adding more links if needed. The removal and adding of a link to a bundle is done by the host by setting a few parameters.

37.22.2.1 Removal of Link

For removing a link from a bundle, the host has to execute the next steps:

- Clear the ML bit in the LMR register placed in the Link Parameter Table.
- Clear the Link entry, according to the Link ID, in all the Classes Extension Tables of the bundle as described in Table 37-16.
- Update the BPT[No_Links] parameter, so now it will be equal to the new no. of links in the bundle

NOTE

When removing a link from a bundle, plain PPP traffic is still processed by the RISC if available on the link. For completely removing the traffic one can change the SI RAM using the shadow RAM.

37.22.2.2 Addition of Link

To add a link in a bundle, the user is required to execute the next steps:

- Prepare the LPT structures, by setting the LPT(BNDLPTR) parameter
- Update the BPT[No_Links] parameter, so now it will be equal to the new number of links in the bundle.
- For each of the classes which are active in the bundle do the following:
  - Read the CPT[Rx MaX] value and put this values to CPT[RxCLx_SQ] of the appropriate entry which correspond to the Link ID of the link which is currently added.
  - Set the Valid bit in this entry. See 37.13.4 for description of each entry.
- Set the LMR[ML] bit.

NOTE

Adding a link to a bundle assumes there was a link that was previously removed from the bundle. i.e. the system has allocated memory for the data structures needed for fragmentation process.

37.23 MURAM Usage

37.23.1 Summary

Table 37-45 summarizes the PPP microcode's usage of multi-user RAM.

Table 37-45. RAM Usage Table

Structure	RAM Usage (bytes)	Pointer Location	Remarks
LPT	268	—	Free area for temporary parameters = 64 bytes
BPT	128	—	—
CPT	128	—	—
CPT Extension	32	CPT	—
Link Statistics Table	48	LPT	—
RxIB	32	LPT	Pointed by Rx_IB_ptr
Tx_LWFQ_ptr	64	LPT	—
Int_TxQD	16	LPT	Only if using external Tx QD. Pointed by Int_TxQD_ptr.
LInt_Frg	64	LPT	Pointed by Tx_LInt_Frg_Ptr
Int_RxQD	32	LPT	Only if using external Rx QD or Rx DeMUX. Pointed by Int_RxQD_ptr.
TxFBPT_BASE	16	BPT	Tx Free Buffer Pool parameter table.
RxFBPT_BASE	16	BPT	Rx Free Buffer Pool parameter table.
CLS Convert Table	16	BPT	Pointed by CLS Convert Table ptr.
CWFQ	64	BPT	—
Packets Reconstruction Management	64	MCC global parameter RAM	—
Packets Reconstruction Table	2 *Packet Reconstruction request entries	PRM	—
SB_mngt_tbl	32	BPT	Pointed by ShB_mngt_tbl_Base.
Int_MLQD	16	CPT	Only if using external Tx QD. Pointed by Int_MLQD_ptr.
Status Ring	nibble*(number of WBD entries)	CPT	—
Demux Management Table	64	DeMUX Page	—
Mux Management Table	56 + (16 if using Ext TxQD)	MUX Page	—

Table 37-45. RAM Usage Table (continued)

Structure	RAM Usage (bytes)	Pointer Location	Remarks
Int Queue parameter Table	32* number of interrupt queues	—	—
MUX parameter Table	64* number of Muxed queues	MMT	—

37.23.2 Example

To determine multi-user RAM usage, consider the following example. Assume a system with 2 bundles, 4 links per bundle, External QDs, and 8 classes per Bundle. Also assume that the number of fragments under the WBD ring is 200 (this means that the Status Ring Base allocates 100 bytes per CPT; see Section 37.16.1, "WBD Ring and Status Ring Size"). Also assume external Demux Fragment Table entries.

The usage of the multi-user RAM would then be as follows:

1. 2 (BPT) * 128= 256 bytes (0x100)
2. 4 (LPT) * 256 * 2 (# bundle) = 2048bytes (0x800)
3. 8 (CPT) * 128 * 2 (# bundle) = 2048 bytes (0x800)
4. 8 (CPT Ext) * 32 *2 (# bundle) = 512. (0x200)
5. 8 (LInt_Frg) * 64 = 512 bytes (0x200)
6. 8 (#links) * (Link Statistics Table) 64= 512 (0x200)
7. (SB_mngt_tbl) * 32* 2 (# bundle)= 64 bytes (0x40)
8. 8 (# links) * (LWFQ) * 64 = 512 bytes (0x200)
9. 8 (# links) (Int_TxQD_PTR)* 16= 128 bytes (0x80)
10. 8 (#links) (RxIB)*32= 256 bytes (0x100)
11. 8 (#links) (RxQD_PTR)*32=256 bytes (0x100)
12. 2 (# bundle) (CWFQ) * 64= 128 bytes (0x80)
13. 2 (# bundle) (FBPT_BASE) * 32 = 64 bytes (0x40)
14. 2 (# bundle) (FPTB)*16 = 32 bytes (0x20)
15. 2 (# bundle) (CLS Convert Table) * 16= 32 bytes (0x20)
16. (Packets Reconstruction Table Base) * (32 req entries*2) = 64 bytes (0x40)
17. (Packets Reconstruction Management) = 64bytes (0x40)
18. [8 (# Class) (Status Ring)*100]* 2(# bundle) = 1600 bytes (0x640)
19. [8 (# Class) (Int_MLQD)*16]* 2(# bundle) =256 bytes (0x100)
20. (Demux Management Table) * 64 = 64 bytes (0x40)
21. (Mux Management Table) * 56 = 56 bytes (0x38)
22. 2*7 (#Interrupt queue parameter table)* 16= 224 bytes (0xE0)

Therefore, the sum for total multi-user RAM usage for this example would be ~ 10.1K bytes.

Chapter 38
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 MCC and UCC. Figure 38-1 shows the structure of the SAM.

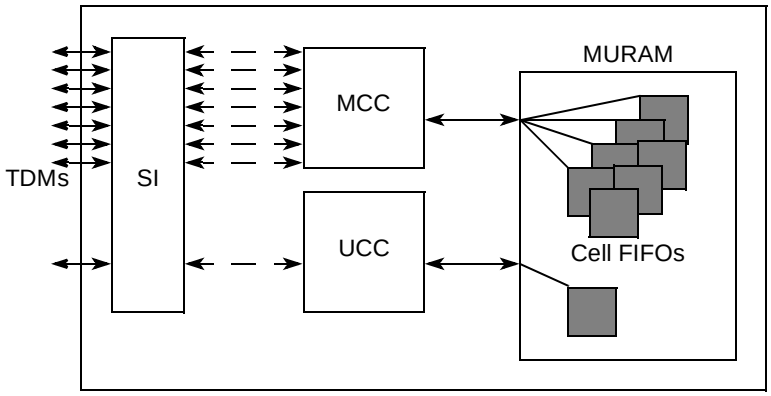


Figure 38-1. Serial ATM Microcode Structure

Each MCC controls 8 TDMs, the MCC can control more TC layers than TDMs, each ATM-based TDM channel has its own cell FIFO ring. Each UCC controls one TDM link and therefore one TC layer. The routing of cells to either an MCC or UCC based TC layer is programmed in the SIRAM.

38.1 Features

The main features of the SAM include the following:

- Operates on both MCC and UCC
- Runs concurrently with MCC SS7, HDLC and Transparent channels
- Eight TDM channels routed by the SI to 64 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

38.2 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 MCC and/or 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 38-2.

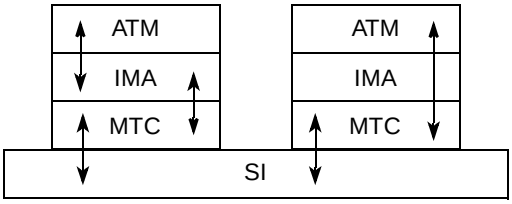


Figure 38-2. QUICC Engine ATM Stack with MTC

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.

38.2.1 Transmit MTC

The transmit MTC architecture is illustrated in Figure 38-3.

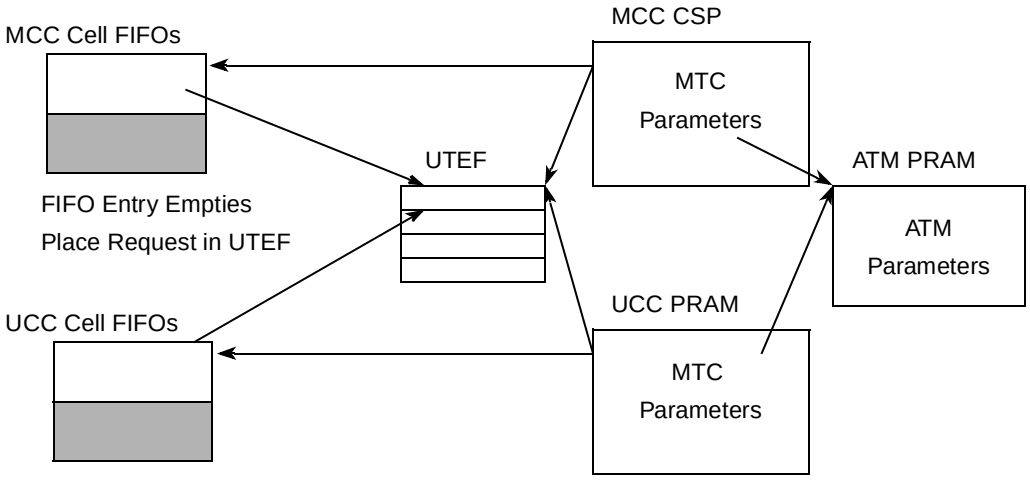


Figure 38-3. MTC Transmit Architecture

Each MURAM-based cell FIFO ring is unique to the MTC controlling the ring. The size of the Tx cell FIFO ring is programmable. Regardless of the QUICC Engine peripheral controlling an MTC, the programming model is compatible. The MTC is independent of the QUICC Engine peripheral on which the cells are generated.

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 or MCC 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 38-3 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 UCC1-8.

38.2.1.1 MCC Transmitter

When an MCC channel is ATM-based, the MCC Tx channel must be programmed as a super channel. The MCC HDLC, SS7, PPP and transparent channels can coexist alongside ATM-based MCC channels. Also, the super channel table must be configured in the MURAM.

When the MCC controls an MTC, the MTC parameters reside in the channel-specific parameters (CSP) for the MCC super channel configured to operate as an MTC layer. The MCC transmit super channel number used for any MTC after subtracting 2 must be divisible by 4. 16-bit super channels should be used

for SAM-based MCC channels, which yields the best overall performance for the SAM when the MCC is used.

The MCC begins transmitting the cell by accessing the current entry in the cell FIFO. It transmits one byte at a time, and the MTC passes it four bytes of the cell payload from the current cell FIFO entry—except when the HEC byte is passed to the MCC and the MTC passes only one byte to the MCC. After the MTC passes all the cell payload to the MCC 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 MCC from the entry to which the incremented MTC_TX_CF_XP to the MCC 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 is delivered before the MCC transmitts the cell present at the incremented MTC_TX_CF_XP. Otherwise, an underrun condition occurs.

38.2.1.2 UCC Transmitter

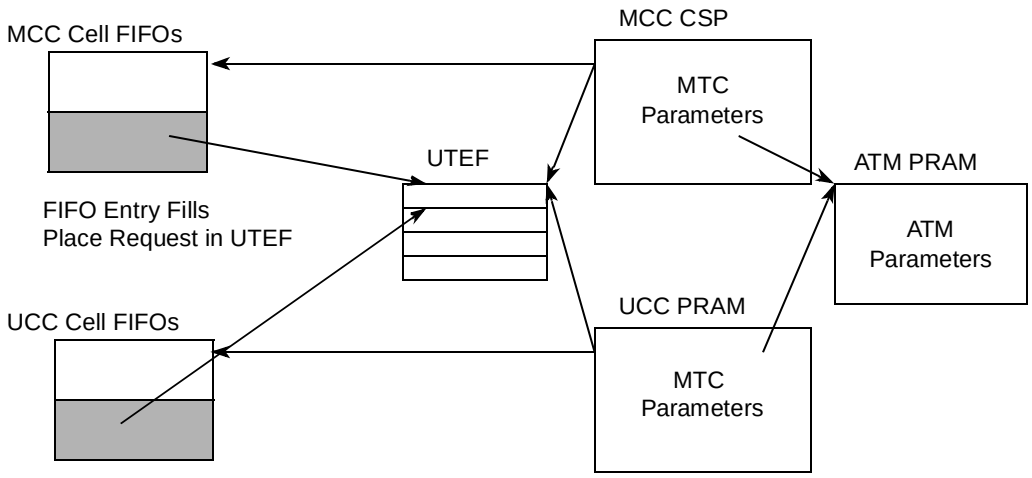
When a UCC controls an MTC, the UCC Parameter RAM should be used to hold the MTC parameters. The UCC transmitter scheme is similar to that of the MCC, except that the UCC transmits four bytes at a time save the HEC byte. The MTC behaves in exactly the same way as the MCC, passing four bytes of the cell payload from the current cell FIFO entry until all of the cell is transmitted.

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.

38.2.2 Receive MTC

The receive MTC architecture is illustrated in Figure 38-4.

Figure 38-4. 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 UCC1-8, so the SNUM used for the ATM controller must be UCC1-8.

38.2.2.1 MCC Receiver

When the MCC controls the MTC, the CSP contains the MTC parameters. For the receiver, no super channel should be enabled for any ATM-based MCC channels. The SIRAM entries corresponding to the ATM-based MCC channels must have the SUPER bit cleared. MCC HDLC, SS7, PPP and transparent channels can coexist alongside ATM-based MCC channels.

Each MCC channel works on 2 bytes at a time, and after 4 bytes are received they are passed to the MTC. The MTC performs the ITU-T I.432 cell delineation on the 4 bytes by checking 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 MCC are used to start building a new cell in the FIFO ring. The ATM microcode processes the fully received cell, and when cell processing is complete it releases this cell FIFO entry by incrementing MTC_RX_CF_XP. The cell processing must complete before the MCC receives another cell in entirety or an overrun condition occurs.

38.2.2.2 UCC Receiver

When a UCC controls an MTC, the UCC PRAM should be used to hold the MTC parameters. The UCC transmitter scheme is similar to that of the MCC, except that the UCC receives works on four bytes at a time and passes this directly to the MTC. The MTC behaves in exactly the same way as the MCC, passing 4 bytes of the cell payload to the current cell FIFO entry until all of the cell is received.

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.

38.2.3 SAM With The Serial Interface

The serial interface defines which timeslots on each of the TDMs are passed to which QUICC Engine peripheral and, for the MCC, which MCC channel. It offers a very flexible TDM mapping scheme for ATM data. It is possible to achieve the following with the SAM:

- More than 1 ATM stream on any given TDM
- More TC layers than TDMs

Figure 38-5 illustrates how to use the TDM to achieve both of these goals:

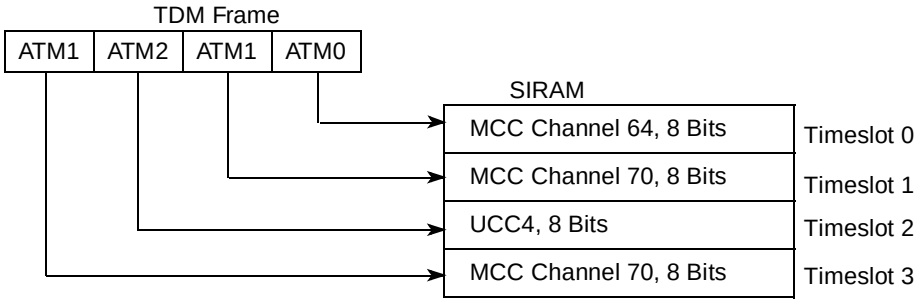


Figure 38-5. SIRAM Flexibility with SAM

The timeslots of one TDM frame can be routed to multiple MCC channels and/or UCCs. Figure 38-5 depicts a TDM stream with four timeslots and three ATM streams. ATM stream 0 in timeslot 0 is routed to MCC channel 64, which is programmed to be an ATM channel (MTC). Similarly, ATM stream 1 in timeslots 1 and 3 is routed to MCC channel 70, which is also an ATM-based MCC channel. Finally, ATM stream 2 on timeslot 2 is routed to UCC4, which is programmed to operate as an MTC.

To have more TC layers than TDMs, each MCC channel active in the SIRAM can be configured as an ATM channel in the CSP for that channel. Up to 64 MTCs can be supported, each requiring 256 bytes of MURAM, which is equivalent to 4 CSP [4*64] worth of MURAM. For example, for an application that requires 16 TC layers running at E1 rates with only 8 TDMs available in the system, the solution is to overclock each TDM at 2*E1 rate and pass 2 ATM streams on each of the 8 TDMs. Use the SIRAM to route each ATM stream to a unique MCC channel programmed to operate as an MTC.

38.2.4 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 38-6. 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.

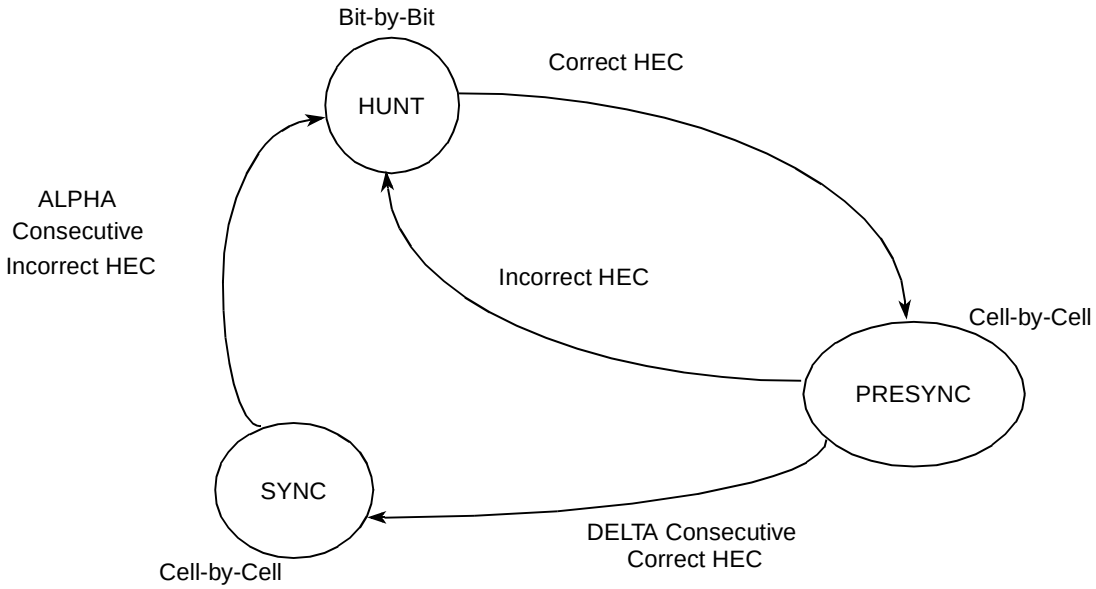


Figure 38-6. TC Cell Delineation State Machine

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 38-7) 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.

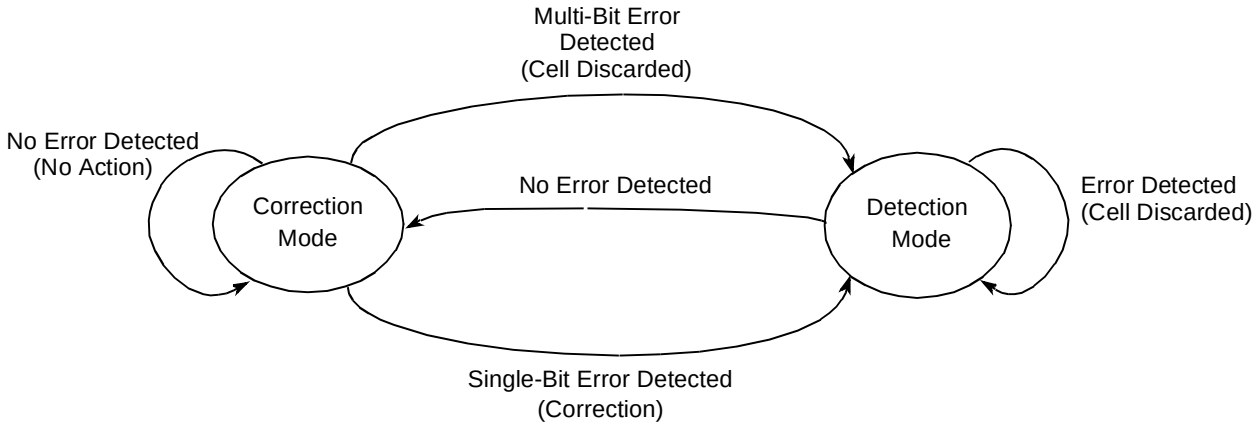


Figure 38-7. HEC: Receiver Modes of Operation

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.

38.3 SAM Programming Model

The MTC parameters are stored in the multi-user RAM (MURAM), Figure 38-8 shows the relationship between internal and external parameters controlled by an MTC:

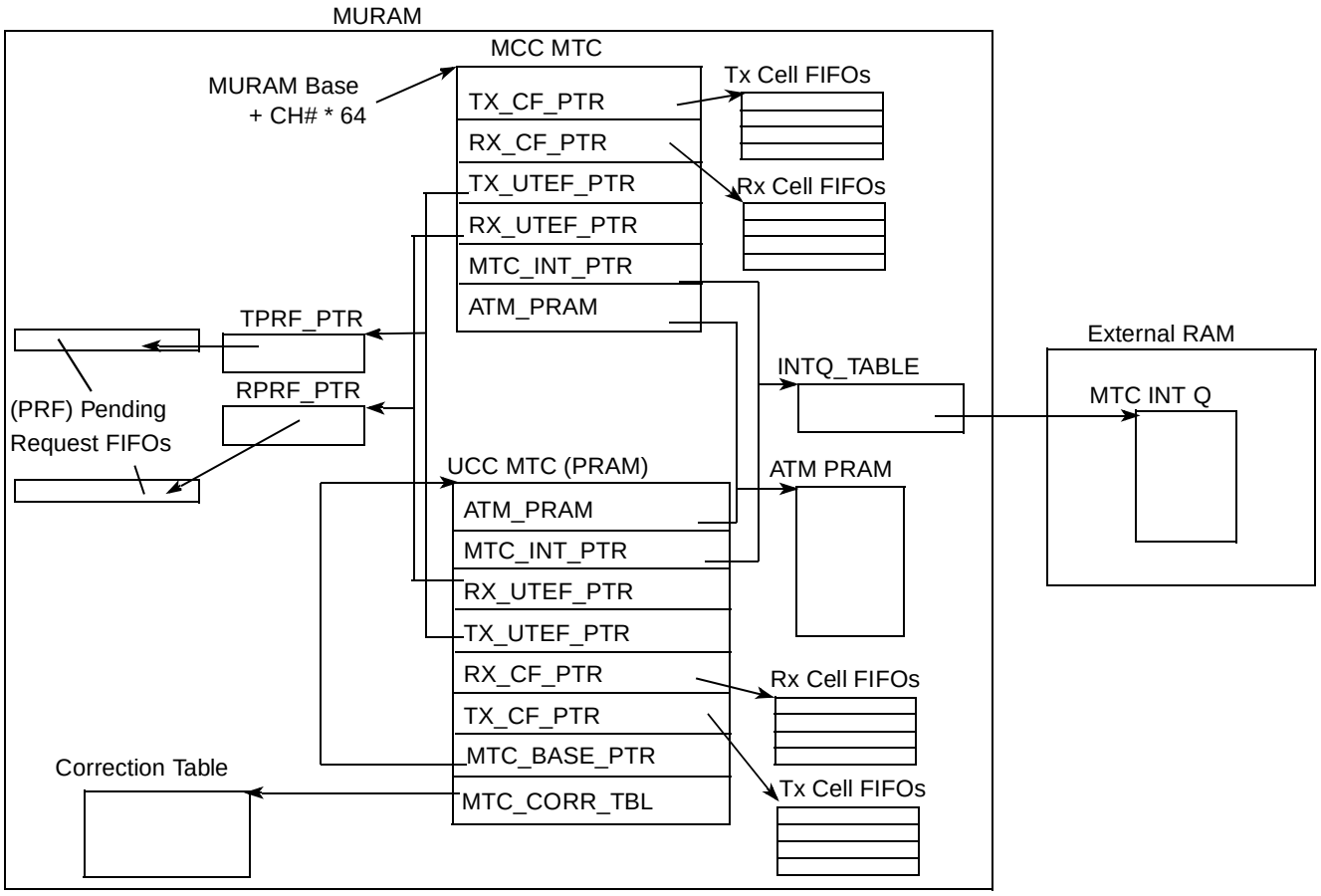


Figure 38-8. MTC Parameters

Each MTC PRAM requires 256 bytes of MURAM, in addition the MTC PRAM must be 256 bytes aligned, when using an MCC channel to hold the MTC Parameters the channel number after subtracting 2 must be a multiple of 4. The base of an MTC PRAM for the MCC is found by using the MCC channel number multiplied by 64 and adding to the MURAM base. The programming model is almost identical when using an UCC or an MCC, the UCC contains an extra parameter, MTC_UCC_BASE_PTR, located in the UCC PRAM. 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 38-8 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. Tthe 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.

38.3.1 MTC PRAM

Table 38-1 shows the MTC PRAM.

Table 38-1. MTC Parameter RAM Map

Offset[1]	Name	Width	Description
0x00	MTC_MODE	Hword	MTC Mode. See Section 38.3.1.1, “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 38.3.1.5, “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 can’t be corrected. This counter is only active when the cell delineation is in the SYNC state.

Table 38-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x14	MTC_STATE_TX	Hword	MTC State Transmit. Initialize to 0x8800. See Section 38.3.1.2, "MTC_STATE_TX Register."
0x16	MTC_STATE_RX	Hword	MTC State Receive. Initialize to 0x0000. See Section 38.3.1.3, "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 38.3.1.5, "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 38.3.1.7, "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 38.3.1.5, "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 38.3.1.5, "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 38.3.1.5, "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 38-1. 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 38.3.1.8, "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 38.3.1.9, "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 38.3.1.10, "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 38.3.1.11, "MTC Transmit Utopia Emulation FIFO."
0x74	MTC_TX_HEC	Byte	MTC Transmit HEC. Microcode managed parameter initialize to 0.
0x75-0x7F	—	—	Reserved. Initialize to 0.

Table 38-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x80	MTC_TSTATE	Word	MTC Transmit State. When using the MCC to control any MTC the MCC channel number associated with an MTC must be aligned to the MTC PRAM base + 128 bytes. For example if the MTC PRAM begins at the offset for MCC channel number 4 (0x100) then MTC_TSTATE must reside at 0x180. Because the MCC expects this parameter to reside at 0x180 and not 0x100, the MCC channel number programmed in the SIRAM should be 6 and not 4. This is true for both the transmitter and receiver. Microcode managed parameter ; Initialize to 0x00800000 regardless of the peripheral used for this MTC (MCC or UCC).
0x84	MTC_ZISTATE	Word	MTC zero-insertion machine state. Microcode managed parameter Initialize to 0x10000207.
0x88	MTC_ZIDATA0	Word	MTC zero insertion high word data buffer. Microcode managed parameter Initialize to 0xFFFFFFFF.
0x8C	MTC_ZIDATA1	Word	MTC zero insertion low word data buffer. Microcode managed parameter Initialize to 0xFFFFFFFF.
0x90-0x99	—	—	Reserved. Initialize to 0.
0x9A	MTC_CHAMR	Hword	MTC Channel Mode Register. This parameter is only valid when the MTC is controlled via an MCC channel, it controls the MCC channel mode of operation. When using the UCC this parameter is reserved and should be set to 0. See Section 38.3.1.4, “MTC Channel Mode Register MTC_CHAMR.”
0x9C-0x9F	—	—	Reserved. Initialize to 0.
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 38.3.1.12, “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. Used when the receiver is the MCC. 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.

Table 38-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
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 38.3.1.13, "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 38.3.1.14, "MTC Receive Multi PHY Address—MTC_RX_MPHY_ADD and MTC_RX_MPHY_ADD_EXT."
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 38.3.1.14, "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 38.3.1.15, "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.

Table 38-1. MTC Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
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 38.3.1.16, “MTC Correction Table—MTC_CORR_TBL.”
0xE4-0xFF	—	—	Reserved. Initialize to 0.

1 Offset from MCC base + (MCC channel number * 64) + 128 when using MCC. Offset from UCCx PRAM when using UCC and MTC_BASE_PTR = UCCx PRAM Base.

38.3.1.1 MTC Mode Register MTC_MODE

Each MTC layer block is configured using a TC layer mode register MTC_MODE, as shown in Figure 38-9.

	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 38-9. MTC Mode Register MTC_MODE

Table 38-2 describes the MTC_MODE fields.

Table 38-2. 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.

Table 38-2. MTC_MODE Field Descriptions (continued)

Bits	Name	Description
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.
15	THD	Transmit HEC disable. 0 Transmit HEC 1 HEC transmission disabled.

38.3.1.2 MTC_STATE_TX Register

MTC_STATE_TX, shown in Figure 38-10, holds the current state of the MTC transmitter. It is required for transmit state fields that require coherency in a multi-RISC environment.

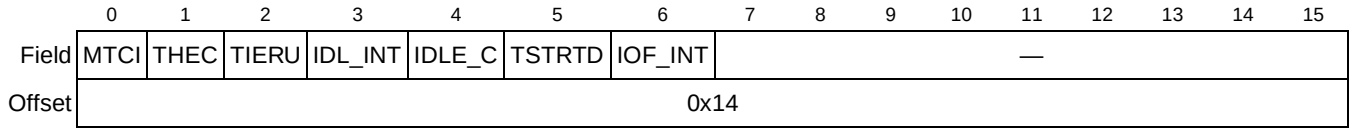


Figure 38-10. MTC_STATE_TX Register

Table 38-3 describes the MTC_STATE_TX fields.

Table 38-3. MTC_STATE_TX Field Descriptions

Bits	Name	Description
0	MTCI	MTC Transmit Idle. Reserved for internal use. Initialize to 1.

Table 38-3. MTC_STATE_TX Field Descriptions (continued)

Bits	Name	Description
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.

38.3.1.3 MTC_STATE_RX Register

MTC_STATE_RX, shown in Figure 38-11, 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 38-11. MTC_STATE_RX Register

Table 38-4 describes the MTC_STATE_RX fields.

Table 38-4. 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.

Table 38-4. MTC_STATE_RX Field Descriptions (continued)

Bits	Name	Description
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.

38.3.1.4 MTC Channel Mode Register MTC_CHAMR

When an MTC is controlled by an MCC channel, the MTC_CHAMR, shown in Figure 38-12, is used to determine whether the channel is an ATM-based channel or HDLC, transparent or SS7 channel.

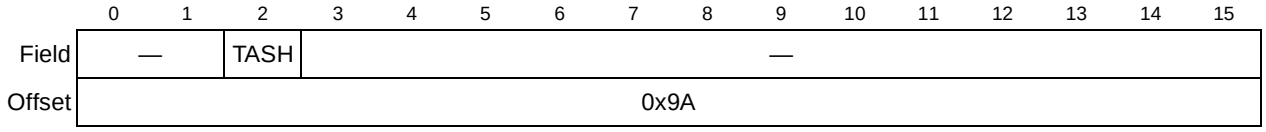


Figure 38-12. MTC Channel Mode Register MTC_CHAMR

Table 38-5 describes MTC_CHAMR fields.

Table 38-5. MTC_CHAMR Field Descriptions

Bits	Name	Description
0–1	—	When MTC_CHAMR[TASH] = 1, these bits are configured according to the description of the MCC controller CHAMR. When MTC_CHAMR[TASH] = 0 initialize theses bits to 0.
2	TASH	Transparent ATM SS7 or HDLC mode. 0 This channel is ATM-based. 1 This channel is not ATM-based. This channel is either an HDLC, Transparent, or SS7 MCC channel.
3–15	—	When MTC_CHAMR[TASH] = 1, these bits are configured according to the description of the MCC controller CHAMR. When MTC_CHAMR[TASH] = 0 initialize these bits to 0.

38.3.1.5 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 38-6 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, For example, a transmit MTC on MCC1 cannot be programmed to use the same MTC interrupt table as a transmit MTC on any UCC. However, multiple transmit MTCs on the same MCC can access the same transmit interrupt table or multiple receivers on the same MCC can access the same receive interrupt table.

Table 38-6. 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 or MCCE this MTC Interrupt Table Uses. See Section 38.3.1.6, “Event Registers” for more details.
0x0D-0x0F	—	—	Reserved. Initialize to 0.

1 Offset from MTC_INT_PTR + MTC_INT_TOF * 16. Must be 16 bytes aligned in the MURAM.

Figure 38-13 shows the format of an interrupt entry written to the MTC Interrupt Queue.

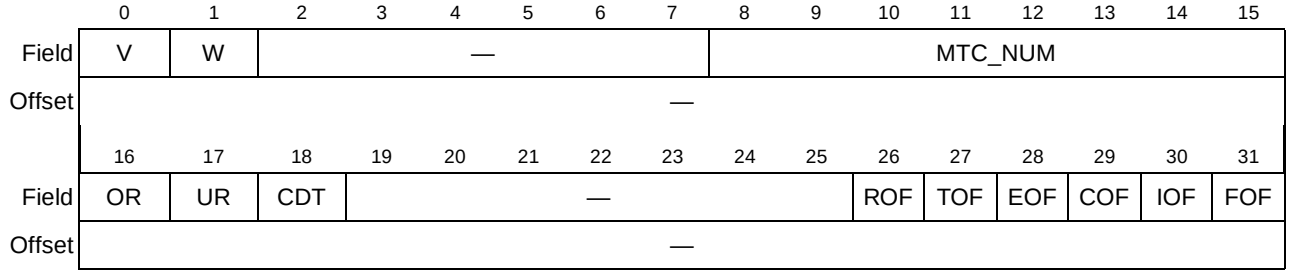


Figure 38-13. MTC Interrupt Queue Entry

Table 38-7 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 38-7. 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.
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.

38.3.1.6 Event Registers

The MCC Event and Mask Register is shown below in Figure 38-14. The MTC_INTE_CTL is used to determine which bit in the event register is set. The corresponding MCCM bit must also be set if the interrupt is to be issued to the core.

	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	TQOV	TINT	GUN	GOV
MTC_INTE_CTL	0x0	0x0	0x1	0x1	0x2	0x2	0x3	0x3	0x4	0x4	0x5	0x5	—	—	—	—

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE	QOV	MTCE
MTC_INTE_CTL	0x8	0x8	0x9	0x9	0xA	0xA	0xB	0xB	0xC	0xC	0xD	0xD	0xE	0xE	0xF	0xF

Figure 38-14. MCC Event/Mask Register

Table 38-8 details the MTC event register interrupt bits.

Table 38-8. MTC Event Register For MCC

Bits	Name	Description
0, 2, 4, 6, 8, 10, 14, 16, 18, 20, 22, 24, 26, 28, 30	QOV	Queue Overflow. When this bit is set the interrupt queue corresponding to the MTC_INTE_CTL setting has experienced an overflow.
1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25, 27, 29, 31	MTCE	MTC Event. The interrupt queue corresponding to the MTC_INTE_CTL has experienced an interrupt event.
12	TQOV	Transmit Queue Overflow. TQOV is set (and an interrupt request is generated) by the QUICC Engine module whenever an overflow occurs in the transmit circular interrupt table. This condition occurs if the QUICC Engine module attempts to write a new interrupt entry into an entry that was not handled by the user. Such an entry is identified by V = 1.
13	TINT	Transmit interrupt. When TINT = 1, at least one new entry in the transmit interrupt circular table was generated by MCC. After clearing it, the user reads the next entry from the transmit interrupt circular table and starts processing a specific channel's exception. The user returns from the interrupt handler when it reaches a table entry with V = 0.[1]
14	GUN	Global Underrun. The MCC Transmit FIFO has experienced an underrun. The MCC should be disabled via the SIMxR and reinitialized.
15	GOV	Global Overrun. The MCC Receive FIFO has experienced an overrun. The MCC should be disabled via the SIMxR and reinitialized.

[1] When an MCC channel is ATM based the CUN, the MCC generates the CUN interrupt in the external MCC interrupt queue for the transmitter. Therefore, the Global MCC Parameters, TINTBASE, TINTPTR and TINTTMP must be initialized when the SAM is in use.

For MCC SS7, HDLC and Transparent channels, the interrupt queue or queues for these channels should not correspond to an MTC_INTE_CTL setting. For example, if these MCC channels are using interrupt queue 0, MTC_INTE_CTL should not be cleared.

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 38-15 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 38-15. UCC Event/Mask Register

Table 38-9 describes the MTC event register interrupt bits.

Table 38-9. 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.
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.

38.3.1.7 MTC Interrupt Queue SDMA Control

Figure 38-16 shows configuration information for the MTC_SCTL.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		—		BIB
Offset	0x24							

Figure 38-16. MTC_SCTL—MTC Interrupt Queue SDMA Control Register

Table 38-10 describes the MTC_SCTL bit fields.

Table 38-10. 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.

38.3.1.8 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 38-17 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 38-17. MTC_TX_ATM_PRAM

Table 38-11 describes the fields of the MTC_TX_ATM_PRAM.

Table 38-11. 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 0x0050 UCC6, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0051 0x0060 UCC7, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0061 0x0070 UCC8, MTC_RX_ATM_PRAM[UR_SNUM] must be programmed to 0x0071

[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.

38.3.1.9 Transmit Cell FIFO Ring

A transmit cell FIFO ring is unique to one MTC. The transmit cell FIFO ring configuration is shown in Figure 38-18.

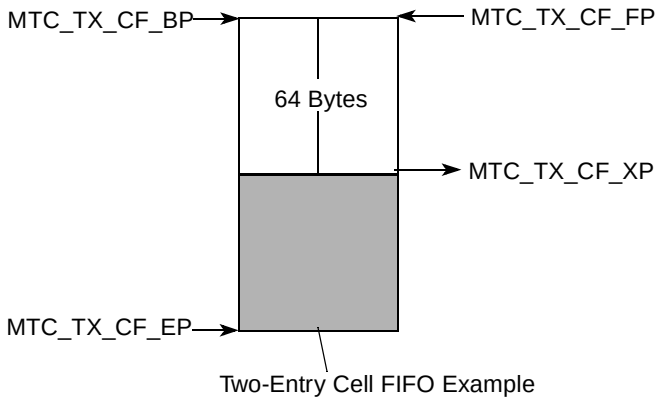


Figure 38-18. MTC Transmit Cell FIFO Ring

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.

38.3.1.10 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 38-19 shows the fields of the MTC_TX_MPHY_ADD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—	IMART	—					1	0	DC		TX_MPHY[4:0]				
Offset	0x64															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		1	—			MPHY_EN	—								
Offset	0x66															

Figure 38-19. MTC_TX_MPHY_ADD Register

Table 38-12 describes the MTC_TX_MPHY_ADD fields.

Table 38-12. 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.
16-17	—	Reserved, initialize to 0.
18	—	Must be initialized to 1.
19–21	—	Reserved, initialize to 0.
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.

38.3.1.11 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 38-13. 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.

1 Offset from MTC_TX_UTEF_PTR. Must be 16 bytes aligned in the MURAM.

The user must clear the PRF during initialization.

38.3.1.12 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 38-20 shows the fields of the MTC_RX_ATM_PRAM.

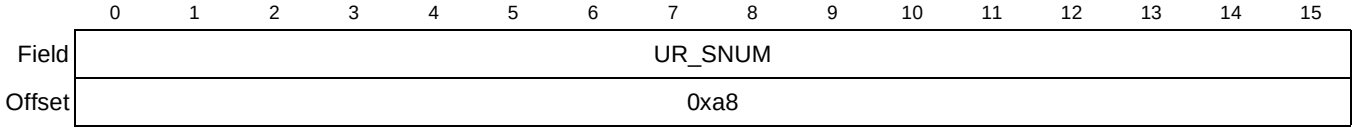


Figure 38-20. MTC_RX_ATM_PRAM

Table 38-14 describes the MTC_RX_ATM_PRAM fields.

Table 38-14. 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 0x0051 UCC6, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0050 0x0061 UCC7, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0060 0x0071 UCC8, MTC_TX_ATM_PRAM[UT_SNUM] must be programmed to 0x0070

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.

38.3.1.13 Receive Cell FIFO Ring

A receive cell FIFO ring is unique to an MTC. The receive cell FIFO ring configuration is shown in Figure 38-21.

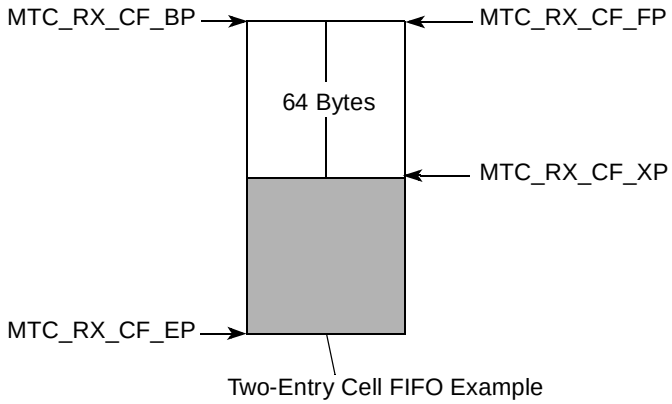


Figure 38-21. MTC Receive Cell FIFO Ring

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.

38.3.1.14 MTC Receive Multi PHY Address—MTC_RX_MPHY_ADD and MTC_RX_MPHY_ADD_EXT

MTC_RX_MPHY_ADD, shown in Figure 38-22 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.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RX_MPHY[3:0]				—											
Offset	0xC0															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—															
Offset	0xC2															

Figure 38-22. MTC_RX_MPHY_ADD Register

Table 38-15 describes the fields of the MTC_RX_MPHY_ADD register.

Table 38-15. 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 38-23 shows the MTC_RX_MPHY_ADD_EXT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—	IMART	—					MPHY_EN	—	1	—					
Offset	0xC4															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	DC		RX_MPHY[4]	0x2f											
Offset	0xC6															

Figure 38-23. MTC_RX_MPHY_ADD_EXT Register

Table 38-16 describes the MTC_RX_MPHY_ADD_EXT fields.

Table 38-16. 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.
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.

38.3.1.15 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 38-17. 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.

1 Offset from MTC_RX_UTEF_PTR. Must be 16 bytes aligned in the MURAM.

38.3.1.16 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 38-18 shows the configuration requirements for the correction table.

Table 38-18. 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

Table 38-18. MTC Correction Table (continued)

Offset[1]	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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.

38.4 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 38-24 shows the IMACNTL.

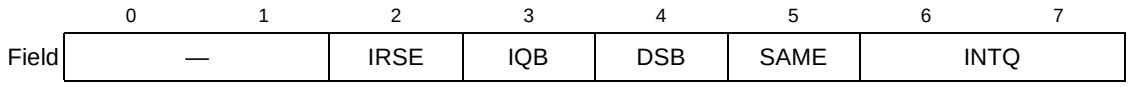


Figure 38-24. IMA Control (IMACNTL)

Table 38-19 describes the IMACNTL bit fields.

Table 38-19. 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.

Table 38-19. IMACNTL Field Descriptions (continued)

Bits	Name	Description
5	SAME	Serial ATM Enable. 0 The TC layer is not SAM based 1 The TC layer is SAM based, i.e. 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.

38.5 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.

38.6 MCC Initialization

If no MCC SS7, HDLC or transparent channels are used in the MCC, the CUN interrupt can still occur. Software must configure the MCC transmit interrupt queue parameters in the Global MCC PRAM.

To configure an MCC channel to act as a MTC the following initialization, perform the following steps:

1. Configure Parallel I/O for the TDM interface required.
2. Initialize the SIU interrupt controller to mask or enable MCCE interrupts as appropriate.
3. Initialize the MCC registers, MCCFx.
4. Initialize the SIRAM and SIRAM registers except SIxGMR.
5. Initialize the MCC global parameters. If no SS7, HDLC, PPP or transparent channels are active in the QUICC Engine block, then the SCTPBASE and associated super channel table should be initialized. Also, the TINTBASE, TINTPTR and TINTTMP should be initialized for the CUN interrupt. For better overall performance, use 16-bit super channels for ATM-based MCC channels that are part of a super channel. On the receiver, only one MCC channel is required per MTC; each SIRAM Rx entry should contain this channel number.
6. Initialize MTC PRAM.
7. Issue the MCC Init Host Command for all channels that are part of the MTC, including any channels programmed in the SIRAM to be part of a super channel connected to the SAM MCC channel.
8. Enable TDM in the SIxGMR.

38.6.1 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 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 verify 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], then update only the transmit or receive MTC parameters. The MTC_TSTATE, MTC_ZISTATE, MTC_ZIDADTA0 and MTC_ZIDADTA1 are part of MTC, but for the MCC, these should not be updated if the TDM is still enabled in the SIxGMR.

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 check that MTC_RX_CF_XP = MTC_RX_CF_FP.
4. Disable TDM through the SIxGMR. If there is more than 1 ATM stream on the TDM and only 1 ATM stream is to be disabled, do not disable the TDM here. Issue the MCCTSTOP command for the MCC Tx super channel used for this MTC. If this channel is to be assigned to another MCC protocol, MTC_CHAMR[TASH] should remain clear until all CSP for this channel are updated except MTSTATE and MRSTATE. Then MTC_CHAMR[TASH] should be set and MTSTATE and MRSTATE should be updated to their default values for the protocol chosen.

After these steps, the MTC is completely disabled and its PRAM area can be updated or reassigned as required. The MCC channel can also be assigned to another protocol, such as HDLC.

38.6.2 Response to GUN or GOV

A GOV or GUN is a non-recoverable event for the MCC; after this event the following procedure should be performed:

1. Clear MTC_MODE[TXEN] and MTC_MODE[RXEN].
2. Disable all TDMs connected to the MCC that experienced GOV or GUN through the SIxGMR
3. Software must wait for MTC_STATE_TX[MTCI, IDLE_C] = 0 and then wait for MTC_TX_CF_FP = MTC_TX_CF_XP for all ATM-based MCC channels

4. Software must wait for MTC_STATE_RX[CFOV_C] = 0 and then verify that MTC_RX_CF_XP = MTC_RX_CF_FP for all ATM-based MCC channels.
5. Issue the MCC Reset Host command.
6. Re-Initialize MTC PRAM for all MTCs.
7. Issue the MCC Init Host command for all channels that are part of the MTC, including any channels programmed in the SIRAM to be part of a super channel.
8. Re-enable any MTC is to be re-enabled by setting the appropriate bit in the SIxGMR.

38.6.3 Response To CUN

When a channel underrun event occurs, perform the following steps:

1. Clear MTC_MODE[TXEN] for the affected MCC channel. If the channel is part of a super channel, the MCCTSTOP should be issued for the channel that directly relates to the super channels CSP. This is not necessarily the CUN channel.
2. Issue the MCCTSTOP command for the channel that experienced CUN. If the channel is part of a super channel, the MCCTSTOP should be issued for the channel that directly relates to the super channels CSP. This is not necessarily the CUN channel.
3. Issue MCC INIT TX, ONE CHANNEL host command for the affected channel.
4. Software must wait for MTC_STATE_TX[MTCI, IDLE_C] = 0 and then wait for MTC_TX_CF_FP = MTC_TX_CF_XP.
5. Update all the transmit MTC parameters to their default state.
6. Set MTC_MODE[TXEN] to restart the MTC transmitter.

38.7 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 29-3).
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.

38.7.1 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.

38.7.2 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 39
Enhanced MSP Microcode

39.1 Introduction

The QUICC Engine multi-service platform derivative (QUICC Engine EMSP) adds ATM layer functionality, suitable for the implementation of various network applications, such as a Port Controller on Subscriber line cards, ATM cell processing, Cable Modem controllers, ATM Multiplexing and concentrating systems and other multiservice and multiprotocol applications especially in the Branch/Enterprise Dial Access Applications.

A T1/E1 Line Card is depicted in Figure 39-1. Eight T1/E1 ports provide the connectivity to the subscribers, and the Utopia is used to connect to the fabric interface of the multi-service platform box. The QUICC Engine EMSP derivative, performs ATM switching functions on the ingress and egress direction.

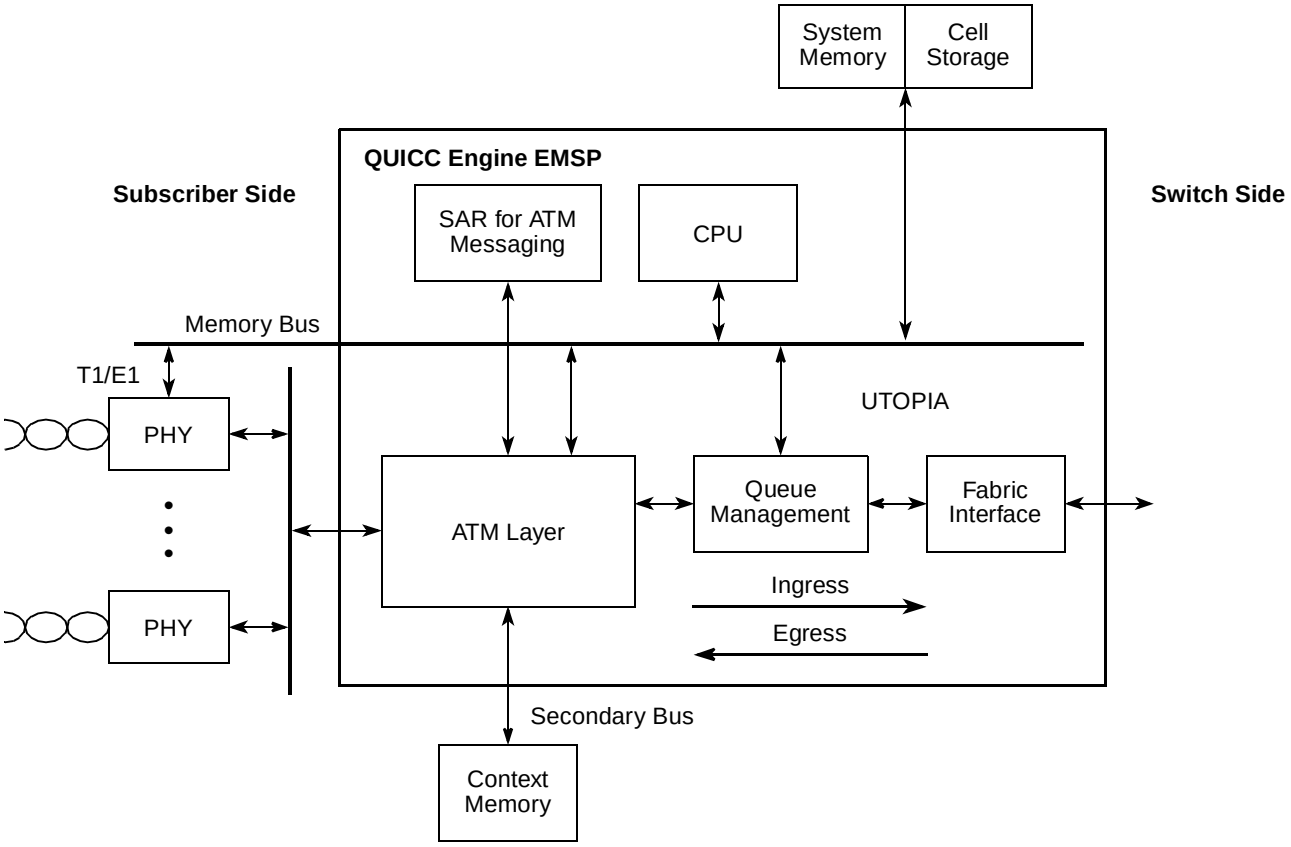


Figure 39-1. Access Line Card Example Using the QUICC Engine EMSP

A DSLAM Access Line Card is depicted Figure 39-2.

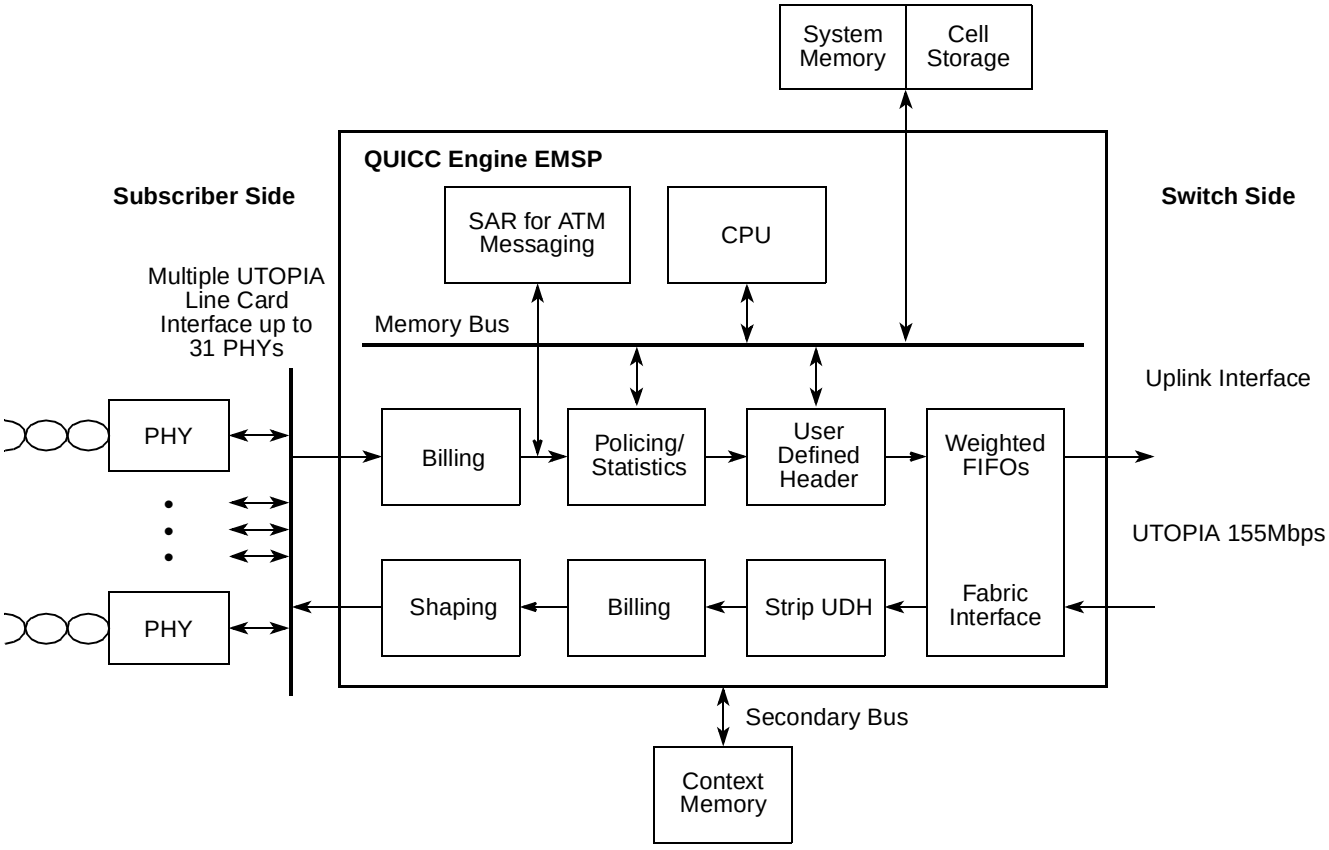


Figure 39-2. DSLAM Access Line Card Example Using the QUICC Engine EMSP

The aggregate bandwidth supported by the QUICC Engine EMSP device depends on the functions enabled in any given configuration. Under certain conditions the device will function as a 155 Mbps switch.

Utopia to Utopia switching functionality is shown in Figure 39-3. An external TC-Layer device performs framing of cells from the T1/E1. Frame-relay, fast Ethernet, and channelized E1/T1 interconnection are provided with the other interfaces on the QUICC Engine EMSP, thus allowing the design of a multi-protocol line card.

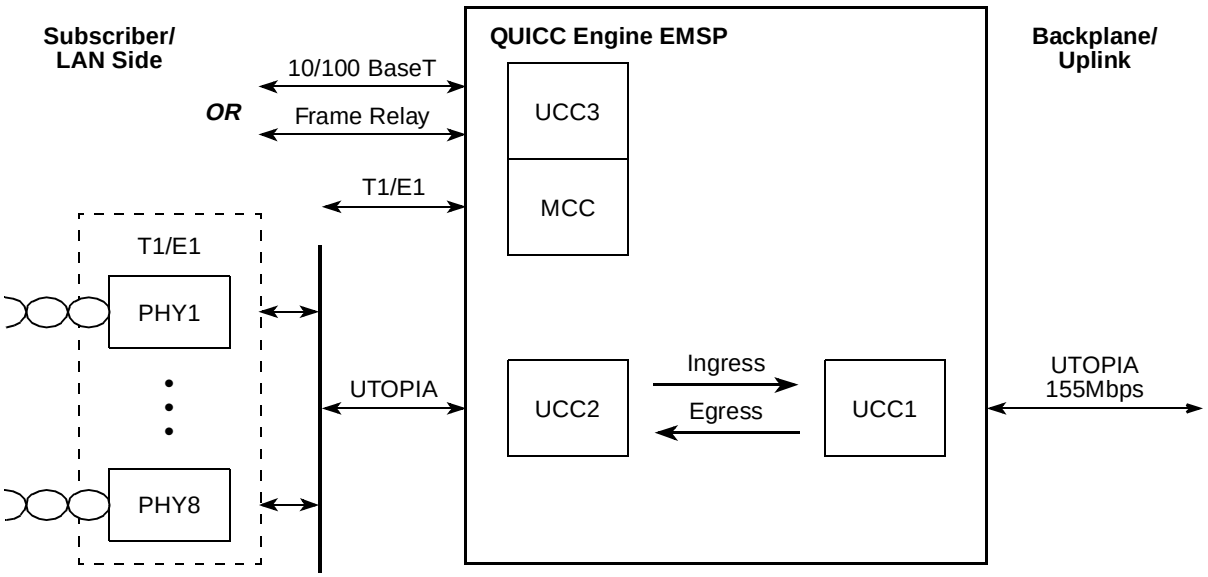


Figure 39-3. General MSP Access Application—External TC Layer

Figure 39-4 shows an implementation of a line card using a single Utopia interface on the QUICC Engine EMSP. The multiphy function on the Utopia interface allows connection to the switch fabric interface, as well as the T1/E1 framer on the same Utopia interface.

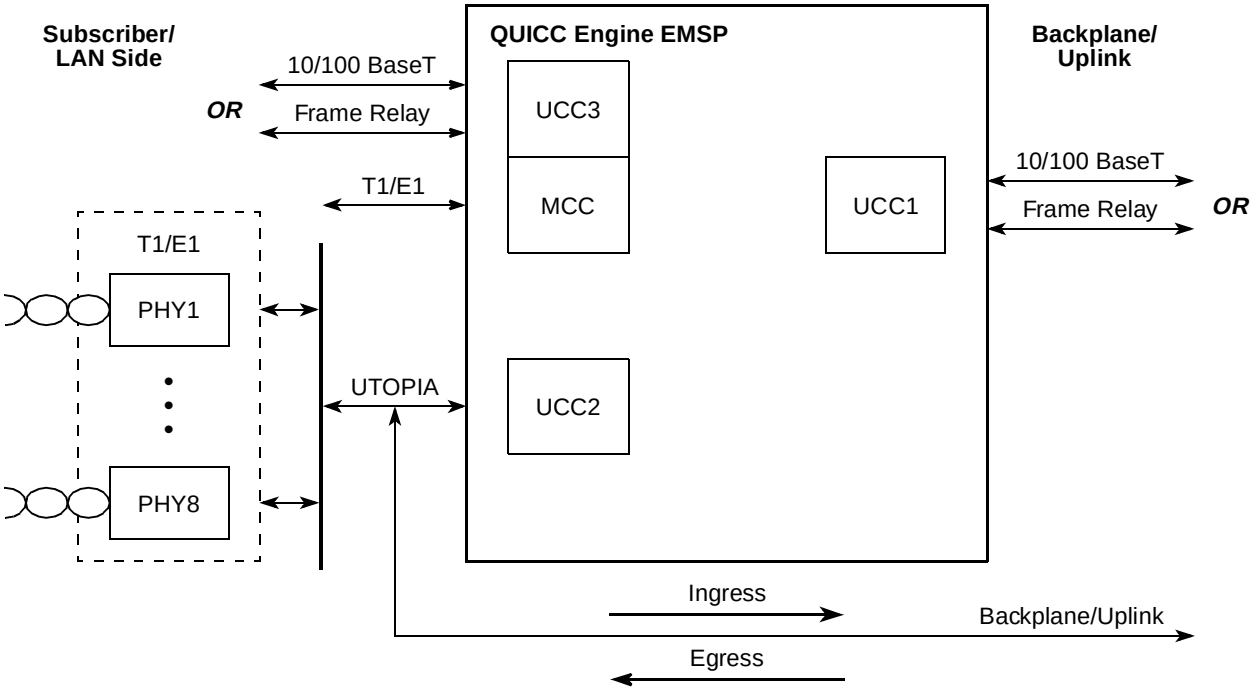


Figure 39-4. General MSP Access Application—Single Utopia

With the QUICC Engine block, it is possible to implement a T1/E1 Dial Access Line card by using the internal ATM TC-layer functionality. The block diagram for such a system is shown in Figure 39-5:

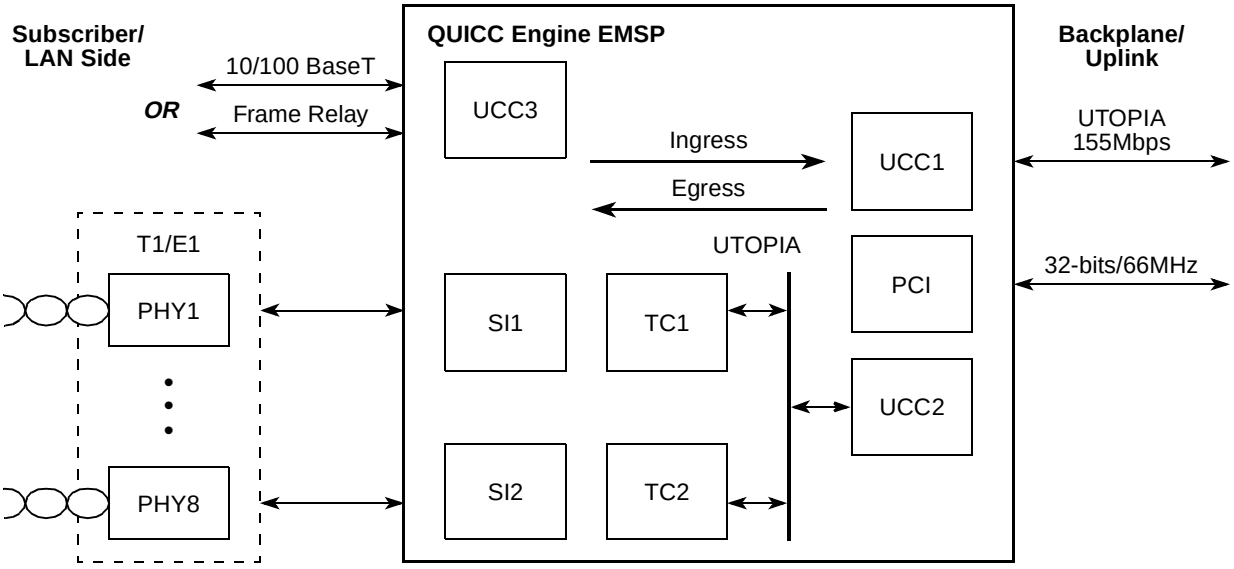


Figure 39-5. General MSP Access Application Internal TC Layer

On the subscriber side, the multi-service platform line card might support a number of different standards like frame relay (1.544 Mbs), T1/E1 (2.048 Mbs), or one or more of the G.lite standards.

On the Backplane side, the MSP transfers the data to the UCC1-Utopia Interface (at 155Mbps peak bit rate). It is possible to add a PCI interface using the internal PCI Bridge. The interface can be used for control/data packets in the system.

Layer 3 (or above) protocol conversion and routing may be done by the internal CPU core, or by an external CPU in the system.

ATM Switching functions like Address Translation, Concentration, Policing and Shaping and Billing are done in firmware without user intervention, except for initial programming and maintenance. ATM cells are SARed on a VC/VP basis, if needed.

The total aggregated bandwidth supported is 8 * 2.048Mbs on the subscriber side, and an average of 8 * 2.048Mb on the Backplane (Utopia interface) side. Peak rate of 155Mbps rate is supported on the Backplane (Utopia interface) side.

The ATM Layer functions which are supported by the QUICC Engine EMSP, and organized in an Access System Line card configuration are shown in Figure 39-6 (Ingress) and Figure 39-7 (Egress).

39.1.1 Ingress Data Flow

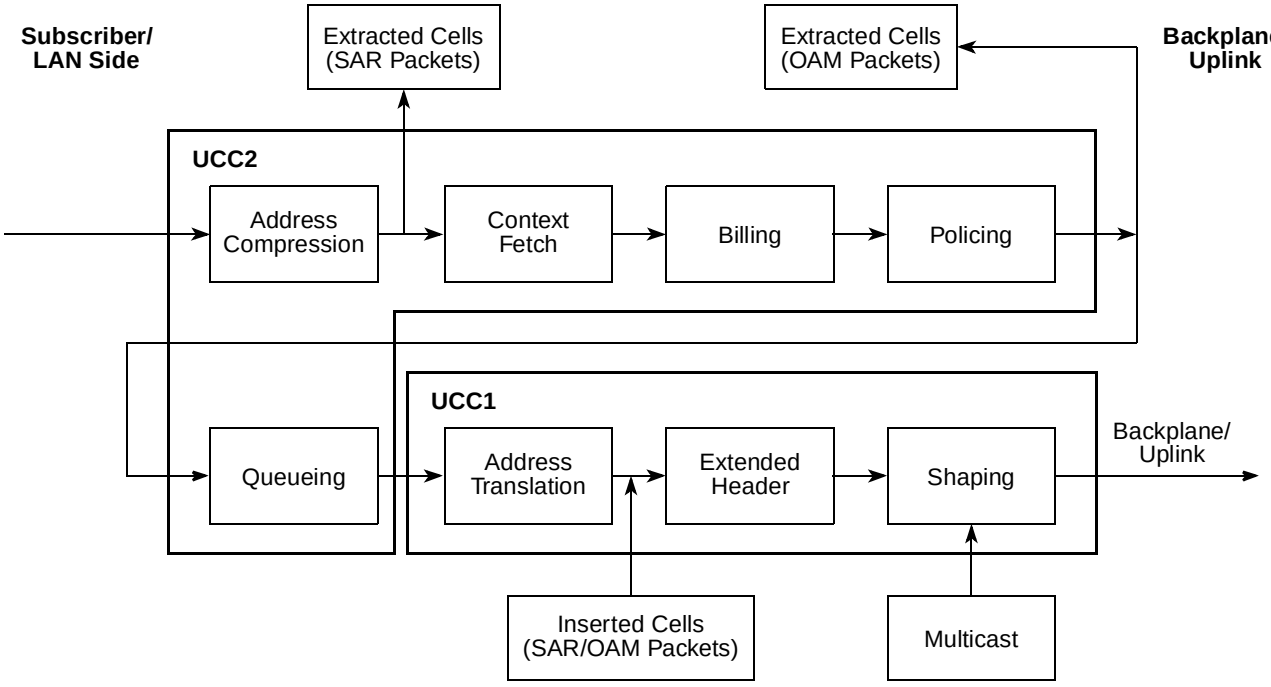


Figure 39-6. QUICC Engine EMSP ATM Layer Functionality—Ingress

Following either PowerQUICC II Pro style Address Compression or Address lookup in a CAM, the connection parameters are fetched. Cell counting is done per connection for billing, and policing operations (if enabled) discard or tags cells or packets. Following address translation, and addition of Switch overhead the cell is placed in a queue for transmission. Cells may be inserted or extracted by the CPU, for OAM management.

Scheduling to the Backplane/Uplink may be done on a per VC basis, or with 4 prioritized FIFO queues. Both mechanisms allow for rate control.

39.1.2 Egress Data Flow

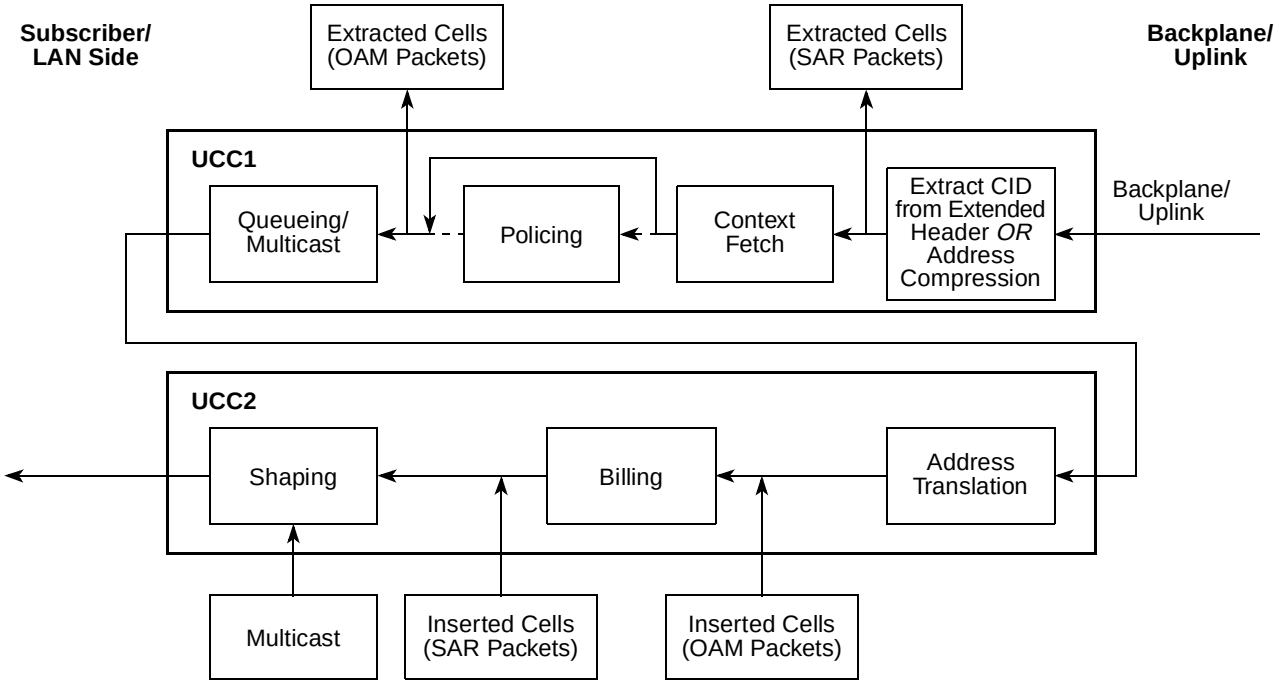


Figure 39-7. QUICC Engine EMSP ATM Layer Functionality—Egress

In the Egress Path, connection context can be defined either by address look-up mechanism or simply can be taken directly from a certain field in a pre-defined user defined header.

The connections are arranged in per VC queues and scheduling is performed with a hierarchical shaper, where VP shaping is performed at top level, and VC shaping at the lower level.

39.2 ATM Switching Functionality

Address Compression, address translation and bundling features of the QUICC Engine EMSP allow for flexible ATM switching features.

39.2.1 VP Switching

This function is supported by ignoring the VCI of the cells. All functions (as policing, queuing, billing, statistics) are performed on a VP basis. Address translation (optionally) alters only the VPI.

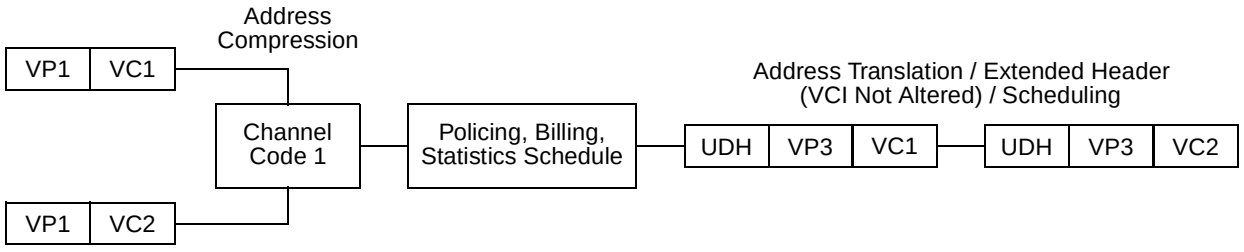


Figure 39-8. VP Switching

39.2.2 VC Switching with Bundling

This feature allows Policing (UPC) on multiple VCIs as one bundle, but perform billing, address translation, Extended Header, scheduling on a per VCC basis.

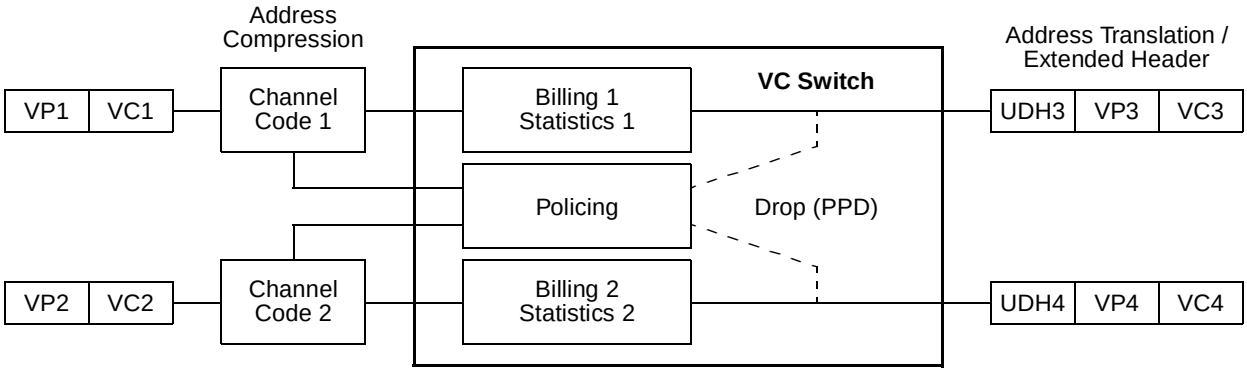


Figure 39-9. VP Switching with Bundling

39.3 Acronyms and Abbreviated Terms

Table 39-1 provides a list of acronyms and abbreviations that are used throughout this chapter.

Table 39-1. Acronyms and Abbreviated Terms

Term	Meaning
APC	ATM pace controller
CAM	Content addresseble memory
CB	Cell body
CLL	Cell linked lists
CLP	Cell loss priority
COV	Cell overhead
EFCI	Explicit forward congestion indication
EPD	Early packet discard
FCP	Free cell pool
FDT	FIFO descriptor table
FTCT	FIFO TCT
GFC	Generic flow control
GFR	Guaranteed frame rate
HTCT	Hierarchical TCT
MPHY	Multi PHY
MRCT	Multicast RCT
MSP	Multi-service platform

Table 39-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
MTCT	Multicast TCT
NCP	Next cell pointer
OAM	Operation and maintenance
PHY	Physical Layer
PM	Performance Managing
PPD	Partial Packet Discard
PTI	Payload Type Indication
QTLS	Queue Thresholds Level Table
RCC	Receive Channel Code
RCT	Receive Connection Table
CETSCR	QUICC Engine Time Stamp Control Register
TCC	Transmit Channel Code
TCT	Transmit Connection Table
SAR	Segmentation And Reassembly
SRCT	Switched Cells Receive Connection Table
STCT	Switched Cells Transmit Connection Table
STCTE	Switched Cells Transmit Connection Table Extension
VC	Virtual channel
VCI	Virtual Channel Indication
VCT	Virtual Channel Table
VP	Virtual Path
VPI	Virtual Path Indication
VPT	Virtual Path Table
UEAD	User Extended Address
UDC	User Defined Cell
UDH	User Defined Header
UPC	Usage Parameter Control

39.4 EMSP Features

- General ATM switching capabilities
  - Integrates ATM layer functionality and PQII SAR Functionality
  - Per VCC/VPC function select (SAR or Switch Functionality)
  - Executes ATM switching between any pair of PHY devices on the same UL2

  - Executes ATM Switching between two UL2 ports
  - UTOPIA Level 2 up to 127 sub-ports for subscriber side
  - 8/16 bit UTOPIA for Fabric connection and Subscriber Connection
  - VPC/ VCC Switching and Address Translation
  - Supports UPC, Queuing, Shaping and statistics gathering per VPC/VCC
  - Queue Management with Congestion Control
  - Supports up to 64K VCCs
  - Buffer Memory (Data Memory) up to 4GB in external DRAM
  - flexible and efficient memory management method, using the concept of **Cell Linked List (CLL)** and **Free Cell Pool**.
  - Address look-up using either external CAM or address compression
  - ATM Forum Compliant Policing per VC or VC Bundle
- Address resolution
  - Algorithmic Address Compression
  - CAM interface (glueless interface to CAM)
  - Channel code extraction directly from Extra Header
- Cells Filtering Options
  - Per VCC/VPC option to copy cells to Egress Port
  - Per VCC/VPC option to copy cells to SAR engine
  - Per VCC/VPC option to copy cells to Raw Cell and Interrupting host
- Billing
  - CLP0 and CLP1 Count per VCC/VPC
  - Interrupt is generated for Billing Counter expiration
- Policing according to ITU-T I.371 and ATM Forum TM Specification 4.1
  - GFR Conformance and GFR Eligibility testing
  - Policing per VCC/VPC and arbitrary VCCs group bundle
  - Two leaky buckets
  - CLP Marking
  - Cell Discard for CLP0, CLP1, or CLP0+1 optionally
  - PPD discard policy is optional for AAL5 VCCs
- Statistics
  - Statistics per VCC/VPC and arbitrary VCCs group Bundle
  - Non conforming CLP0 Counter with optional expiration interrupt generation
  - Non conforming CLP1 Counter with optional expiration interrupt generation
- ATM Header Translation
  - VPI only translation
  - VCI only translation
  - VPI/VCI translation

- Queue Managements
  - Per VCC/VPC Queuing
  - VCCs Bundle Queuing
  - Sets of thresholds per Queue: EFCI marking, EPD discard, CLP1 discard, CLP0+1 discard
  - Supports PPD for CLP1 and CLP0 discard
  - Controllable Service Level for Non-Eligible GFR frames
- Scheduling
  - Per VCC, Per VPC or arbitrary VCCs group Bundle Prioritized FIFO/Weighted Round Robin / Mixed mode Scheduling
  - Merging of SAR Scheduling system and Switch Scheduling system
  - Bandwidth management by a Calendar Wheel Scheduling scheme (APC)
  - Up to eight priority Calendar Wheels per sub-port
  - Hierarchical Scheduling by a combination of APC (Upper Level) and Prioritized FIFO/Weighted Round Robin / Mixed mode for second level
- Traffic Shaping
  - Peak Cell Rate shaping (PCR)
  - Peak and Sustain Rate shaping (PCR, SCR, BT)
  - Peak and Minimum Rate shaping (PCR, MCR)
  - VBR, UBR+ shaping.
- User defined cells
  - Up to 65 bytes. (fabric side) for per VCC queuing and shaping
  - Up to 63 bytes for VCC Bundle Queuing
- Multicast
  - No duplicating cells in memory
  - Multicast through per VCC/VPC queuing system
  - Multicast through WFQ algorithm.
- OAM (I.610)
  - Detection of F4 Segment and End To End OAM for VPC Switching
  - Detection of F5 Segment and End To End OAM for VCC Switching
  - OAM Received cells are either un-touched, Removed from cell flow, or copy to CPU queue
  - OAM Cells are inserted to cell flow by Host i/f
  - Performance Monitoring support without Host intervention
- Host Controllability by Host Commands
  - Establishing and tearing of new queue from FCP Host Commands
- Applications
  - Multiservice Platform for Access Line Card Applications
  - Concentrators
  - Remote Access Units

- DSLAM Applications: Mux/Demux G.lite subscribers
- Cell Processing, ATM switching, Shaping and UPC controller

39.5 Functional Description and Programming Model

39.5.1 Introduction

The QUICC Engine EMSP functionality adds ATM switching functions. In order to achieve high data rates for the switching functions, new data structures are defined for cells received in switching mode.

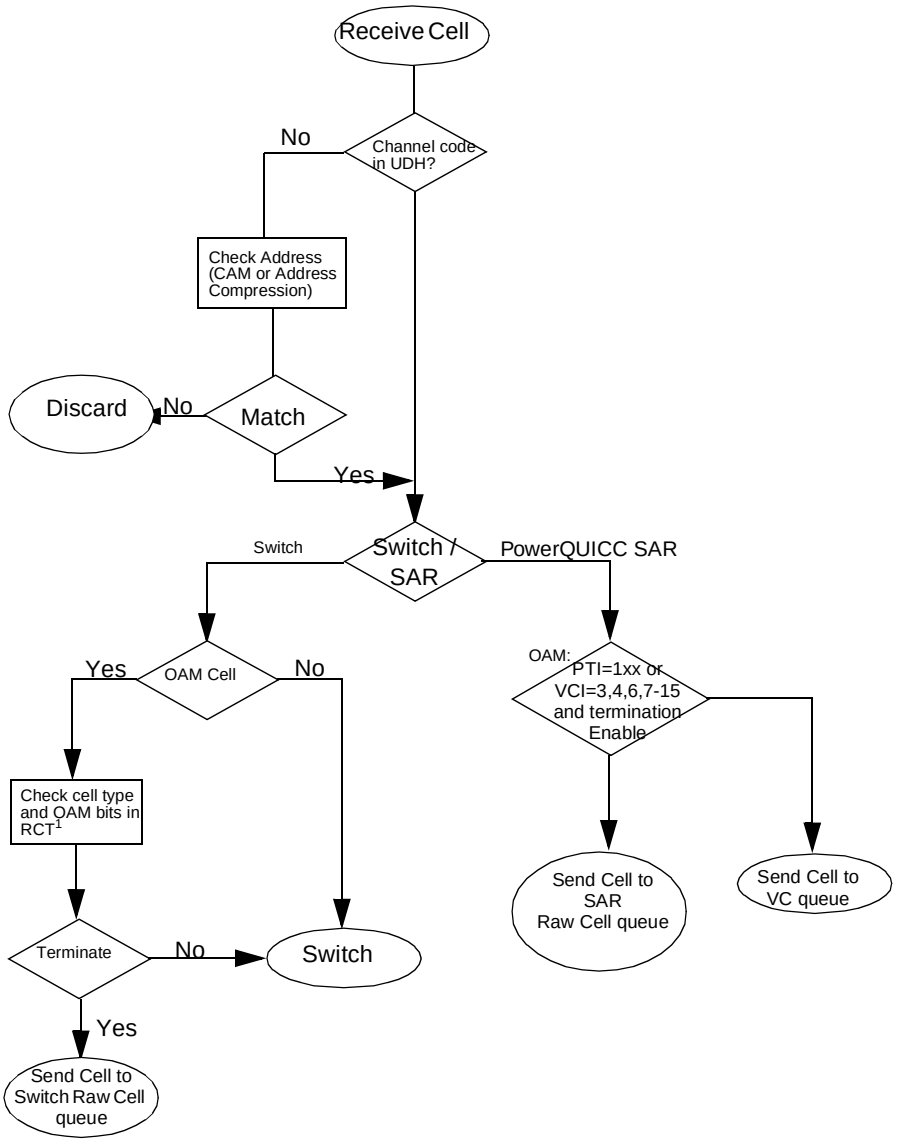
Every cell has an associate channel-code (RCC for arriving cells, TCC for transmitted cells). For incoming cells the channel code can be retrieved in one of three ways (determined in the GMODE register): a) Directly from the extra header which resides before the ATM header on the cell, b) as a result of address compression mechanism performed on the VPI/VCI of the cell, c) as a result of a search in an external CAM. The Channel-code has a connection table entry associated with it (RCT for receive or TCT for transmit). In some cases there are additional parameters in extended tables. In the RCT/TCT AAL=110 indicates that this channel is switched instead of SARed. If switched, the switching data structures are used; if SARed the PowerQUICC II Pro data structures are used. The PowerQUICC II Pro structures are composed of Buffer descriptors-rings which point to Data buffers (which may or may nor be pre-allocated). The switching data structures are composed of cell linked lists (CLLs), and a free cell pool (FCP) of empty linked cells. The Switching function automatically fetches free cells from the free cell pool and returns them after transmission. The following figure (Figure 39-10) shows the flow for SAR or switch modes.

The scheduling mechanism for transmission can be one of two (on a per-PHY basis): Either the PowerQUICC II Pro APC is selected, or a simple eight FIFO system is selected. The PowerQUICC II Pro mechanism has been enhanced to support two level hierarchy; this allows for per VP shaping on the entire link, and per VC shaping for each VC inside the VP. The FIFO mechanism supports three scheduling algorithms: Fixed priority, Weighted Round Robin, and a combination of the two.

Notes about this chapter:

- Figure 39-30 is a general overview of the data structures used for queueing and scheduling in the MSP. This figure can be used as a reference while reading this chapter.
- The term UDC (User Defined Cell), UDH (User Defined Header) and Extra header are used intermittently to describe the cells that have addition bytes prepended to it.

The following figure depicts the flow that distinguishes the PowerQUICC II Pro SARing functionality from the MSP functionality.



OAM Functionality is described in Section 39.15, “OAM Support.”

Figure 39-10. Receive Flow

39.6 Address Resolution

39.7 MSP Buffer Management

The receiver routine in the MSP determines where the cells are routed to. There are three options:

— Cells that are SARed by and stored locally in memory for further processing. Those cells have the RCT[AAL] bits not equal 110. The SARing function is performed by the MPC8260 PowerQUICC II Pro SAR functionality. Some of the SARed cells are stored in the Raw-Cell-Queue

— Cells that are stored in the local Raw cell queue. Those cells may belong to connections which are switched or SARed. Normally those are OAM cells that are terminated in the switch.

— Cells that are switched. Those cells are automatically transferred to the transmitter, after appropriate address translation, and extra header handling has been performed by the RISC. The transmitter automatically sends the cells out the outgoing port. These cells are never handled by the local CPU.

The first two categories of cells are stored in local memory using the QUICC Engine memory model. This model includes Buffer Descriptors rings, that are initialized by the CPU, and data buffers, that may be preallocated, or fetched from a free buffer pool.

The third category is used only for cells that are automatically switched from the receiver to the transmitter. The following discussion pertains only to this category.

The MSP programming model offers a flexible and efficient memory management method, using the concept of **Cell Linked List (CLL)**. The main advantage of this method is it's memory efficiency - Memory allocation is done on a global basis. The user does not have to allocate memory per a connection, or per a specific queue.

The User allocates memory globally for all connections by defining a "**Free Cell Pool (FCP)**". The FCP is a pool of Empty cells pointers. The size of each empty cell is 64 bytes. Some of the bytes beyond the 52 bytes of the ATM cell are used by the RISC. The cells in the free cell pool may reside at any address in the 4Gbyte memory space.

On the receive side, the MSP automatically fetches free cells pointers from the FCP, and links them to the appropriate queue (either a per-VC queue or one of the FIFO queues). On the transmit side, the MSP automatically return free cells pointers to the FCP. The "movement" of cells is done by updating the pointers to the cells. This memory management method enables adaptive memory allocation for each connection.

39.7.1 Free Cell Pool Definition

The user allocates memory globally for all connections by defining a "**Free Cell Pool** (**FCP**)". Up to two free cell pools are available. The UCC Parameter RAM has an entry called FCP_BASE. This entry contains a pointer to the internal multi user RAM, containing the Free-Cell-Pool Descriptor. The free-Cell-Pool descriptor contains pointers to manage the Free-Cell-Pool. A system having one free cell pool should program the FCP-BASE in all UCC's to the same value.

The free cell pool pointer table (See Figure 39-11) includes sets of pointers of empty cells.

All queues in the MSP Switching function are composed CLL's. The queue chosen for storage is determined by the receive process running on each incoming cell. The empty cells in the FCP contain a 52 bytes space of Cell Body (CB) and 12 bytes space of cell Overhead (COV).

The Receiver pools the next Cell pointer using FCP#_Offset_Out located in the Free Cell Pool Descriptor in the parameter ram. The Transmitter returns the Cell pointer to an entry pointed by FCP#_Offset_In in the same structure.

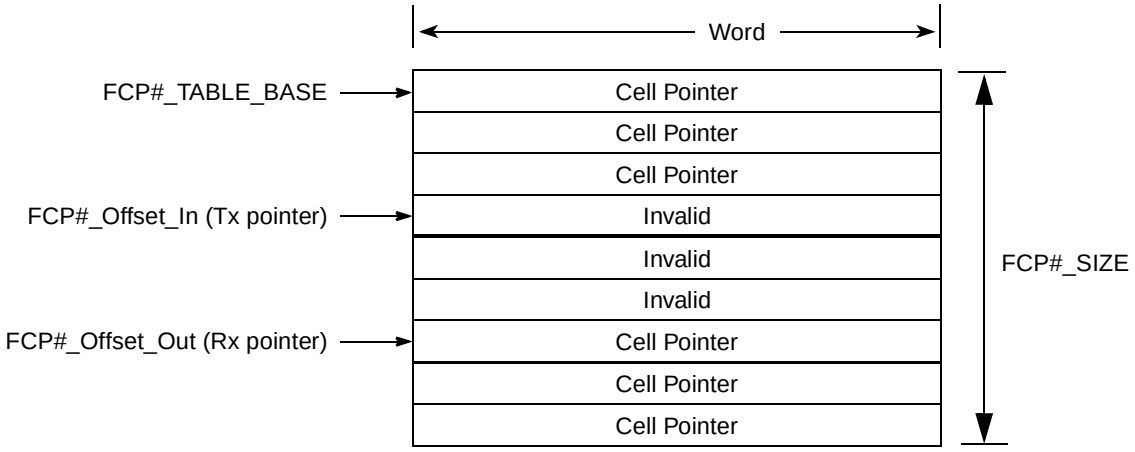


Figure 39-11. Free Cell Pool Pointer Table

39.7.2 Free Cell Pools Descriptor

Two Free Cell Pools are available to the Switch. This allows for more flexible memory allocation for different type of services. The descriptor to the FCPs is located in the multi-user RAM. The base addresses to the descriptor (FCP_BASE) is specified in Parameter RAM for the UCC.

The reserved area is for future microcode expansions.

Table 39-2. Free Cell Pools Descriptor

Offset	Name	Width	Description
0x00	FCP1_TABLE_BASE	Word	Holds the pointer to the first entry in the free cell pool. FCP1_BASE should be word aligned. User-defined.
0x04	FCP1_OFFSET_OUT	Word	Free cell pool Receiver pointer. Offset from the FCP1_BASE to the current entry used by the receiver. The actual address of the current received FCP entry = FCP1_BASE + FCP1_OFFSET_OUT. **Initialize to 0x4**. When the GET command is issued the RISC will take the cell pointer pointed by this entry and will advance the offset to the next entry.
0x08	FCP1_OFFSET_IN	Word	Free cell pool Transmitter pointer. Offset from the FCP1_BASE to the current entry which is being returned to the free cell pool. **Initialize to 0x0**. This implies this entry is empty and available for a returned cell pointer by the transmitter. Note: The actual number of pointers after initialization is less by one. IMPORTANT: The transmitter will never advance this entry so it would match the offset out value. (The RISC will advance it's offset_out value to match offset in and in this case the FCP is fully exhausted a FCPBE interrupt is issued by the RISC). When the PUT command is issued the RISC will return a cell pointer (placed in COMM_INFO field in the parameter ram page) to the FCP entry pointed by offset in and will advance offset in to the next entry.
0x0C	FCP1_RX_TH	Word	When the difference between the Offset_In and Offset_Out cross this value an THRx interrupt is generated. It indicates that the amount of cells left in the system has reached the low watermark.

Table 39-2. Free Cell Pools Descriptor (continued)

Offset	Name	Width	Description
0x10	FCP1_TX_TH	Word	If the difference between Offset_In and Offset_Out cross this value an THTx interrupt is generated. It indicates that the amount of cells in the system has reached the High watermark.
0x14	FCP1_SIZE	Word	User initialized to the Size in bytes of the FCP entries.
0x18	FCP1_IEV	Hword	Interrupt Event Register.(Section 39.7.2.1, “FCPx_IEV Entry”)
0x1A	FCP1_IMASK	Hword	Interrupt Mask Register.(See 39.7.2.2, “FCP_IMASK Entry”)
0x1C-1F			Reserved. Set to zero.
0x20	FCP2_TABLE_BASE	Word	See FCP1_TABLE_BASE description.
0x24	FCP2_OFFSET_OUT	Word	See FCP1_OFFSET_OUT description.
0x28	FCP2_OFFSET_IN	Word	See FCP1_OFFSET_IN description.
0x2C	FCP2_RX_TH	Word	See FCP1_RX_TH description.
0x30	FCP2_TX_TH	Word	See FCP1_TX_TH description.
0x34	FCP2_SIZE	Word	See FCP1_SIZE description.
0x38	FCP2_IEV	Hword	See FCP1_IEV description.
0x3A	FCP2_IMASK	Hword	See FCP1_IMASK description.
0x3C-3F			Reserved. Set to zero.

39.7.2.1 FCPx_IEV Entry

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
THRxE	THTxE	FCPBE													

Table 39-3 describes the fields.

Table 39-3. FCPx_IEV

Bits	Name	Description
0	THRxE	If (FCP_COUNT <= FCP_RX_TH) this event bit is set, and an interrupt is given to CPU if not masked by THRxM bit.
1	THTxE	If (FCP_COUNT > = FCP_TX_TH) this event bit is set, and an interrupt is given to CPU if not masked by THTxM bit.
2	FCPBE	Free Cell Pool Busy Interrupt Event. This event bit is set, and an interrupt is given to CPU if not masked by FCPBM bit.
3-15		Reserved. Set to zero

39.7.2.2 FCP_IMASK Entry

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
THRxM	THTxM	FCPBM													

Table 39-4. FCP_IMASK

Bits	Name	Description
0	THRxM	0- Mask THRxE Interrupt. (THRxE is set, but UCCE[FCPI] bit is not set). 1- enable setting of UCCE[FCPI]
1	THTxM	0- Mask THTxE interrupt (THTxE is set, but UCCE[FCPI] bit is not set). 1- enable setting of UCCE[FCPI]
2	FCPBM	0- Mask FCPBE Interrupt (FCPBE is set, but UCCE[FCPI] bit is not set) 1- enable setting of UCCE[FCPI]
3-15		Reserved. Set to zero

39.7.3 Free Cell Format

The Free cells residing in the Free Cell Pool have the following format:

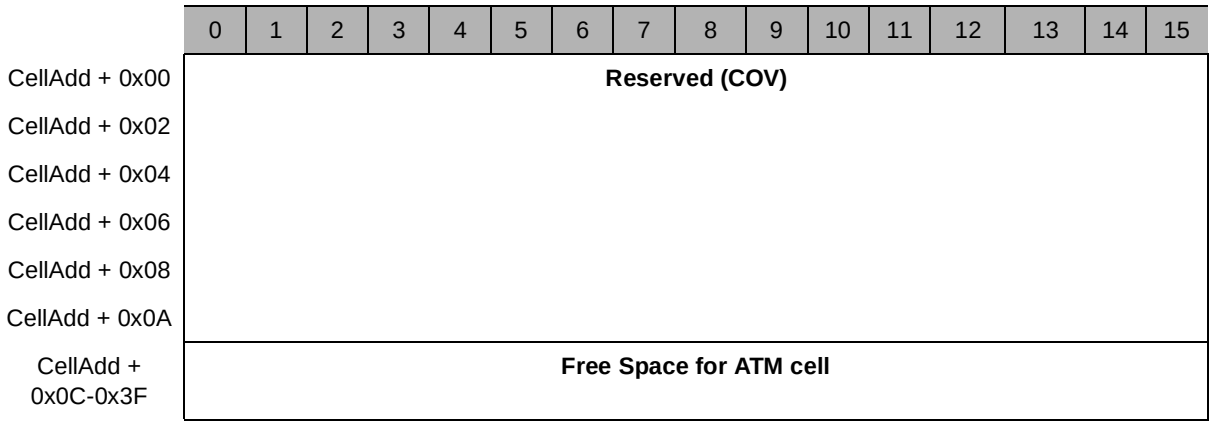


Figure 39-12. Free Cell Format

The user programs the FCP at initialization. Once the MSP starts running the user software must not touch the FCP. The RISC automatically pools and returns cell pointers when it moves the cells around.

39.7.4 Switched Cells Queue definition

The mechanism described below applies to all the queues in the PowerQUICC II Pro MSP. The user initializes at least one Free Cell Pool (FCP) of cell pointers. It is possible to initialize two free cell pools, and allocate cells from one of them on a connection basis. In addition to that for each active queue (either per-VC queue, FIFO queue or Multicast Queue), the user programs the respective TCT and RCT, and allocates one empty cell to the queue. This allocation is done by a special host command see

Section 39.20.1, “MSP FCP Commands” which provides the cell address to the application. The application will initialize RCT[EFDP/QPTRL] and TCT[QPTRLF] or FDTi[FIFO_FIRST] to point to this cell location. Except for this unique cell, all memory allocation is done automatically by the HW during run time. When a channel is being removed from the APC the application can “return” the last remaining cell pointer to the free cell pool by a different host command.

Figure 39-13 and Figure 39-14 show this mechanism as applied in a per-VC queue system. The FCP is full of Empty cell pointers (Number of entries in FCP pointer table -1), as defined previously. The Per VC queue CLL has one empty cell in it. When a cell is received in this connection, one cell pointer is extracted from the FCP and transferred to the Per VC Queue CLL

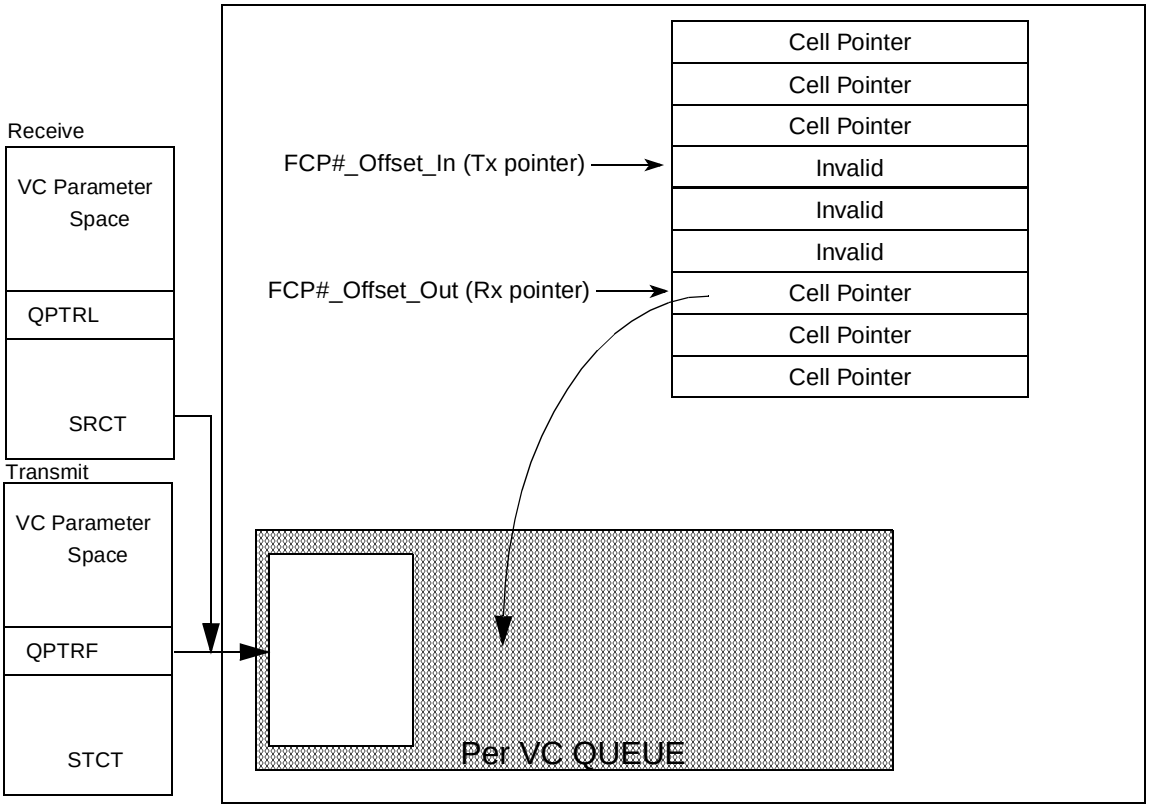


Figure 39-13. Per VC Queue: Initial State

The situation after the cell transfer is shown in Figure 39-14 below.

The first cell pointer from the FCP was extracted and changed to be the last cell in the Per VC Rx Queue Cell-FIFO.

When the scheduler schedules this connection for transmission, the transmitter does the same, but in the opposite direction. It takes the First cell in the Per VC Queue CLL, transmits its data, and returns the cell pointer back to the FCP to an entry pointed by FCP#_Offset_In.

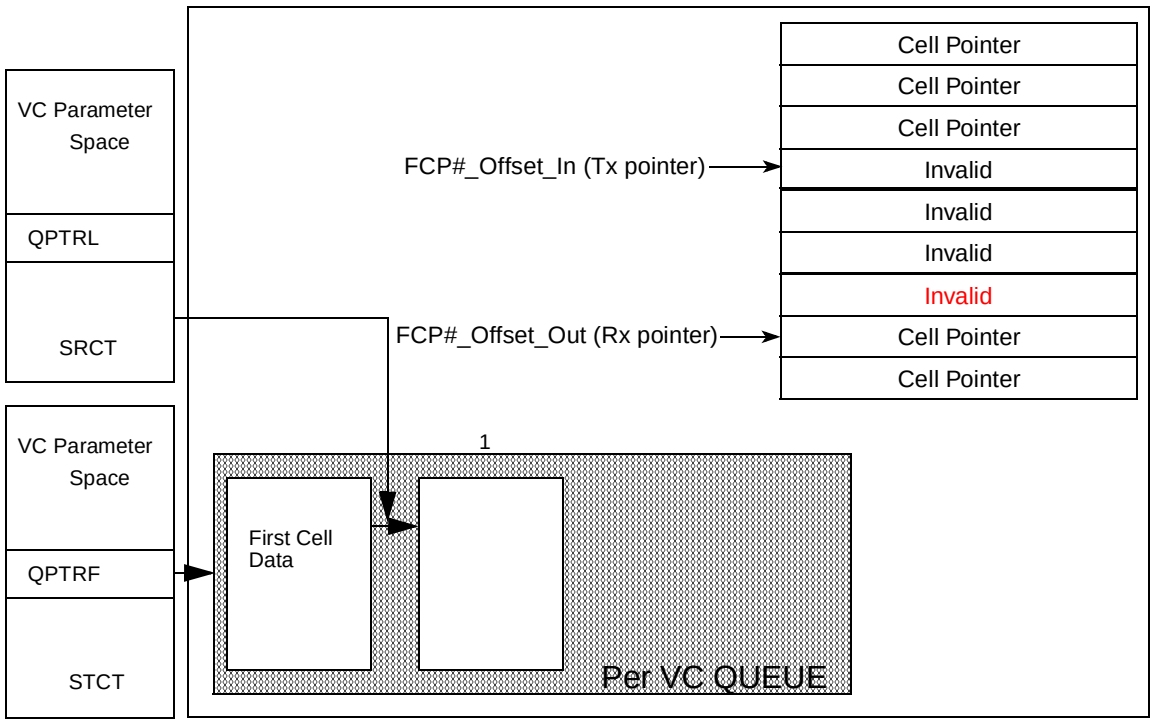


Figure 39-14. Per VC Queue: After Receiving a Cell

In this way, the receiver extracts the cell pointer from the FCP#_offset_out entry, and substitute it as the last cell in the per VC queue CLL. The transmitter extracts the first cell from the Per VC queue CLL, and returns it to an entry pointed by FCP#_offset_in.

When a connection requires more memory than usual, for instance because its transmitter is starving, or its receiver had a moment of peak cell rate, it takes more cells from the FCP. Those cells are returned to the FCP at last after the transmitter completes transmission.

Before pooling a cell from the FCP, the receiver compares FCP#_offset_out to FCP#_offset_in, in the Fifo Descriptor Table (FDT). If they are equal, an Overrun condition occurs, and interrupt is generated to the CPU.

39.7.5 Switched Cells Format

These cells are automatically switched from the receiver to the transmitter. They are organized in CLLs and reside either in VC queues or in FIFOs.

The information in this section is given for informative reasons only, and may be used for debugging purposes. The user software does not access those cells during normal operation.

The cells have the following format:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CellAdd + 0x00	**Next Cell Pointer (NCP)**															
CellAdd + 0x02																
CellAdd + 0x04	**TCC (Transmit Channel Code)**															
CellAdd + 0x06	**Reserved**															
CellAdd + 0x08																
CellAdd + 0x0A																
CellAdd + 0x0C-0x3F	**ATM cell**															

Figure 39-15. Switched ATM Cell Format

39.7.6 CPU cells

Switched cells sent to the CPU (OAM cells), are queued in the raw cell queue which is configured to AAL0 mode. The same queue is used by the QUICC Engine SAR function. The user software distinguishes between the cells by examining bits in the AAL0 RxBD. The extended definition of the AAL0 RxBD follows. For information regarding SEGT, ENDT & VPC of this channel the user should refer to the RCT of the specified channel code.

The BD has the following format:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**E**	-	**W**	**I**			**CM**		OAMERR		CRE	OAM				
Offset + 0x02	Channel Code															
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**															
Offset + 0x06																

Figure 39-16. AAL0 RxBD for OAM cells on switched connections

Table 39-5. AAL0 OAM 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 RISC 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 RISC. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved
	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 RISC 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	—	—
	6	CM	• Continuous mode 0 Normal operation. 1 The RISC does not clear the E bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the RISC next accesses this BD.
	7	—	—
	8	OAMERR	Rx OAM error. Indicates an OAM cell which is erroneous and was removed terminated by the switch. See Table 39-32 for details.
	9	—	Reserved
	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
0x02	—	CC	Channel code. This field functions as Channel code no. which specifies the channel code associated with this OAM cell.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

39.7.7 Multicast Cell Format

For details see Section 39.14.7, “Multicast Cells Format.”

39.8 Queue Management

Each cell queue (VCC, VPC, or any arbitrary bundle of VCCs queue) is associated with a set of 8 thresholds. When the number of cells in the queue exceeds a threshold level, a specific action is taken. Figure 39-20 describes typical threshold setting. The location of the 4 bit threshold field for both Per VCC/VPC queue and VCC bundle queues is the RCT field QTLS

When an incoming cell causes the EFCI threshold to be exceeded, an explicit forward congestion indication is given by the transmitter (The transmitter sets the middle PTI bit of the cell to be transmitted immediately following the reception that caused the EFCI threshold to be exceeded).

Five modes of queue management are available for the user:

c) GFR- This mode is implementing TM4.1. All conditions specified in section 4.5.5.1 for cell conformance definition are checked. Two modes are available for cell's CLP change from 1->0. F-GCRA algorithm is available in the leaky bucket (UPC) implementation. In addition to this, TAGGING and FIFO queue management are implemented as described in Section 39.8.3, "Queue Management for GFR Service."

d) PPD - Partial packet discard. In this mode each cell in a frame is tested for conformance by the UPC (if enabled) and by queue management unit so that it doesn't exceed PPD_CLP1 or PPD_CLP0 thresholds. Once a cell is non-conforming to one of those rules, it is discarded. All following cell in frame will be discarded as well, with no policing at all. The last cell in frame will be transmitted so synchronization won't be lost. If first cell was discarded all the frame including the last cell will be discarded.

e) EPD - Early packet discard. In this mode First cell in frame is tested for conformance by policer (if enabled) and by queue management unit so that it doesn't exceed EPD_CLP1 or EPD_CLP0 thresholds defined by the user. If conformed, all frame will be transmitted. If failed they will be discarded. Once the frame is received the UPC (GCRA) is being updated with no impact on discard decision. When next first cell arrives it is tested for conformance against the GCRA in it's current state.

f) EPD_PPD - Early packet discard on first cell in frame and Partial packet discard on next cells in frame.In this combined mode The first cell in frame is tested for conformance according to EPD mode (Early packet discard) using EPD_CLP1 or EPD_CLP0 thresholds defined by the user. If failed all the frame will be discarded. If the first cell conformed, next cells in the frame are tested according to PPD mode (Partial packet discard) for conformance using PPD_CLP1 or PPD_CLP0 thresholds.

g) Cell based discard mode- In this mode there is no tracking for any frames running over the ATM and the traffic is treated as AAL0 traffic. Each cell is tested for conformance as a "stand alone" cell by the policer and queue management unit using CB_CLP1 or CB_CLP0 thresholds. CB_CLP0 and CB_CLP1 thresholds are used in a non Framed traffic so that the User can differ the thresholds sets between Frame mode traffic and a non Frame mode traffic.

In all of these modes once the queue/FIFO limits are reached i.e. CLP0+1 threshold is exceeded cells will be discarded and PPD mechanism is activated if in AAL5 traffic.

Figure 39-6 summarizes the different thresholds and their induced behavior.

Table 39-6. Queue Threshold Level Set

Name	Description
EFCI	When the number of cells in the queue exceeds this threshold the PTI “middle” bit of the ATM cell header is set.
EPD_CLP0	frames in EPD/GFR mode - When the number of cells in the queue in exceeds this threshold CLP0 cells will be discarded.
EPD_CLP1	frames in EPD/GFR mode - When the number of cells in the queue exceeds this threshold CLP1 cells will be discarded.
PPD_CLP0	frames in PPDmode - When the number of cells in the queue in exceeds this threshold CLP0 cells will be discarded.
PPD_CLP1	frames in PPDmode - When the number of cells in the queue exceeds this threshold CLP1 cells will be discarded.
CB_CLP0	Cell Based mode - When the number of cells in the queue exceeds this threshold CLP0 cells will be discarded.
CB_CLP1	Cell Based mode - When the number of cells in the queue exceeds this threshold CLP1 cells will be discarded.
CLP0+1	When the number of cells in the queue exceeds this threshold all cells associated with this queue will be discarded.

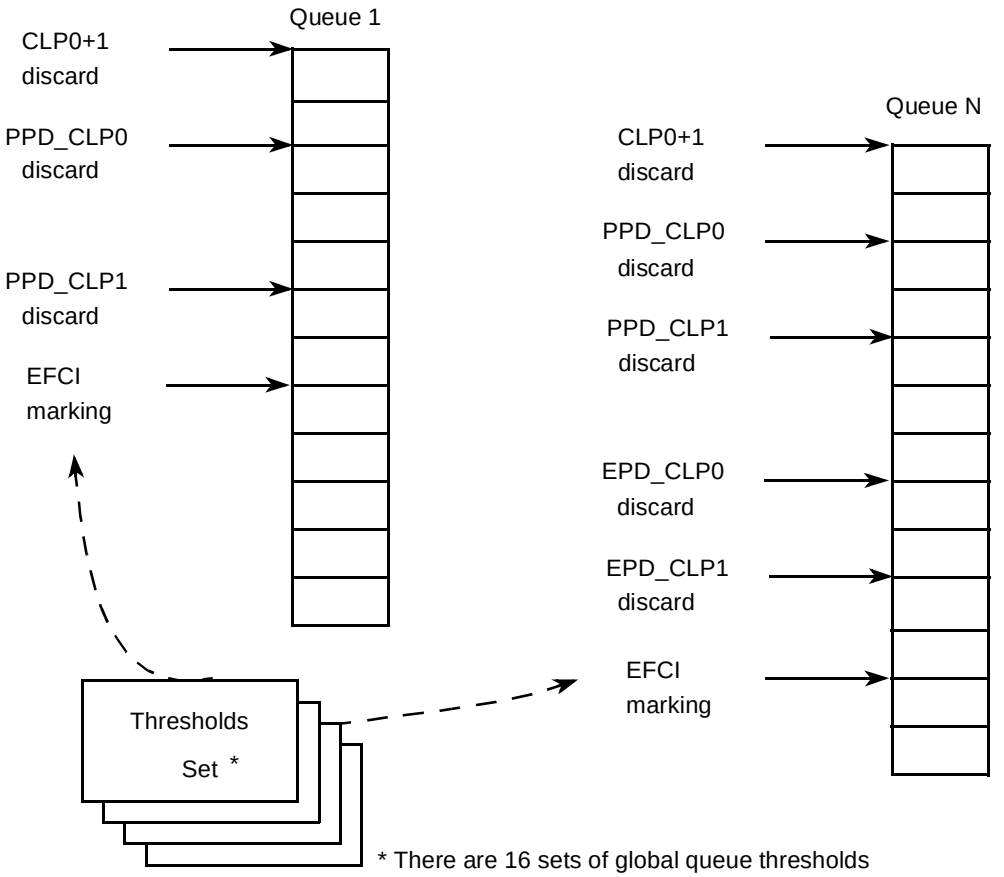


Figure 39-17. Typical Threshold Setting

39.8.1 Queues Threshold Level Table(s)

The Queue thresholds level table includes 16 sets of thresholds levels. Each queue is associated with one set of thresholds. The queue thresholds level tables reside in the multi user RAM. QLTS_BASE entry in the parameter RAM points to the start address of this table. The start address of each thresholds set is: QTLS_BASE + RCT[QTLS]*16.

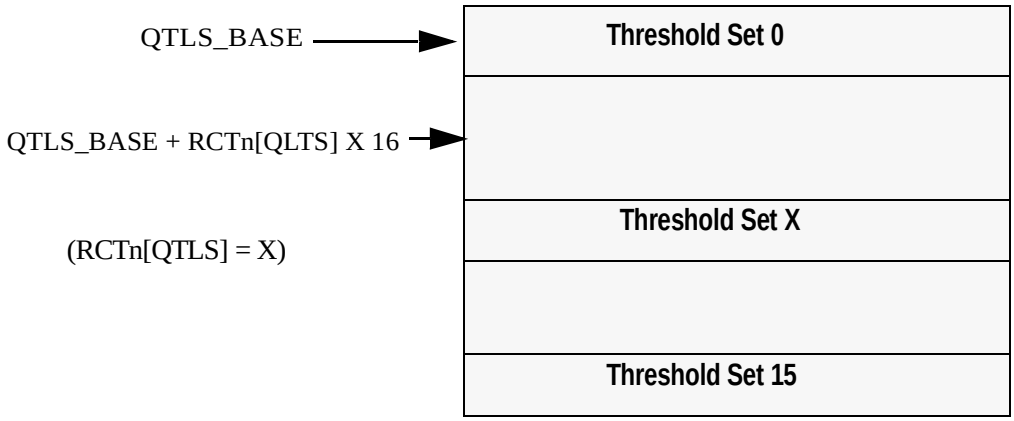


Figure 39-18. Queue Levels Sets Table

Table 39-7. Threshold Sets Memory Location

Address	Name	Width	Description
QTLS_BASE + 00	**EFCI_TH0**	Half Word	EFCI Threshold (Set 0)
QTLS_BASE + 02	**CLP0+1_TH0**	Half Word	CLP0+1 Threshold (Set 0)
QTLS_BASE + 04	**EPD_CLP1_TH0**	Half Word	EPD/GFR CLP1 Threshold (Set 0)
QTLS_BASE + 06	**EPD_CLP0_TH0**	Half Word	EPD/GFR CLP0 Threshold (Set 0)
QTLS_BASE + 08	**PPD_CLP1_TH0**	Half Word	PPD CLP1 Threshold (Set 0)
QTLS_BASE + 0A	**PPD_CLP0_TH0**	Half Word	PPD CLP0 Threshold (Set 0)
QTLS_BASE + 0C	**CB_CLP1_TH0**	Half Word	Cell based CLP1 Threshold (Set 0)
QTLS_BASE + 0F	**CB_CLP0_TH0**	Half Word	Cell based CLP0 Threshold (Set 0)
QTLS_BASE + 10	**EFCI_TH1**	Half Word	EFCI Threshold (Set 1)
QTLS_BASE + 12	**CLP0+1_TH1**	Half Word	CLP0+1 Threshold (Set 1)
QTLS_BASE + 14	**EPD_CLP1_TH1**	Half Word	EPD/GFR CLP1 Threshold (Set 1)
QTLS_BASE + 16	**EPD_CLP0_TH1**	Half Word	EPD/GFR CLP0 Threshold (Set 1)
QTLS_BASE + 18	**PPD_CLP1_TH0**	Half Word	PPD CLP1 Threshold (Set 1)
QTLS_BASE + 1A	**PPD_CLP0_TH0**	Half Word	PPD CLP0 Threshold (Set 1)
QTLS_BASE + 1C	**CB_CLP1_TH0**	Half Word	Cell based CLP1 Threshold (Set 1)
QTLS_BASE + 1E	**CB_CLP0_TH0**	Half Word	Cell based CLP0 Threshold (Set 1)
QTLS_BASE + F0	**EFCI_TH15**	Half Word	EFCI Threshold (Set 15)
QTLS_BASE + F2	**CLP0+1_TH15**	Half Word	CLP0+1 Threshold (Set 15)
QTLS_BASE + F4	**EPD_CLP1_TH15**	Half Word	EPD/GFR CLP1 Threshold (Set 15)

Table 39-7. Threshold Sets Memory Location (continued)

Address	Name	Width	Description
QTLS_BASE + F6	**EPD_CLP0_TH15**	Half Word	EPD/GFR CLP0 Threshold (Set 15)
QTLS_BASE + F8	**PPD_CLP1_TH15**	Half Word	PPD CLP1 Threshold (Set 15)
QTLS_BASE + FA	**PPD_CLP0_TH15**	Half Word	PPD CLP0 Threshold (Set 15)
QTLS_BASE + FC	**CB_CLP1_TH15**	Half Word	Cell based CLP1 Threshold (Set 15)
QTLS_BASE + FE	**CB_CLP0_TH15**	Half Word	Cell based CLP0 Threshold (Set 15)

NOTE

The actual values of the thresholds are limited by the maximum number of cells in a queue. This number is determined by the limitations of the programming model. For example in the STCT the QCC value is limited by 14 bits. therefore a relevant threshold value should be a number between 1 - (2^{14}-1).

The threshold mechanism is supposed to ensure that the Queue cells counters (QCC for STCT and FIFO_COUNT for FIFO), do not overflow; this means that cells should be discarded before these counters reach their maximum value. If these counters overflow, cells are discarded.

39.8.2 Adaptive Threshold Levels

It is well understood that there should be a tight relationship between the FCP Memory utilization and the thresholds the MSP uses for accepting, marking or discarding a cell. Due to the dynamic nature of the system and the varying number of connections it is usually desired to use different sets of thresholds for different levels of FCP memory occupancy. For example, if the number of connections is relatively low, it is generally possible to allocate more Queue resources for each connection by using higher threshold levels. The MSP supports this feature by providing software hook-ups for managing the Threshold levels dynamic behavior using any desired algorithm.

The MSP Free Cell Pool descriptors include two programmable thresholds for indicating the FCP usage. Each of these two thresholds generate a maskable interrupt. One threshold (FCP1_TX_TH) generates an interrupt (if not masked) when the number of cell pointers is equal or larger than the threshold, the other threshold (FCP1_RX_TH) generates an interrupt when the number of cell pointers is equal or smaller than the threshold. This mechanism in conjunction with CPU software allows for the implementation of various schemes of buffer management in the system.

Example for implementation of Hysteresis on the free buffer pool:

At initialization, the transmit threshold (FCP1_TX_TH) associated interrupt is disabled and the other (FCP1_RX_TH) is enabled. When the higher threshold is exceeded the situation reverses, the higher threshold associated interrupt is disabled and the other is enabled, and so on. When the host software detects an interrupt it performs an update of the QTLS table or switches to another table as explained above. This introduces a hysteresis like behavior as depicted in the figure below:

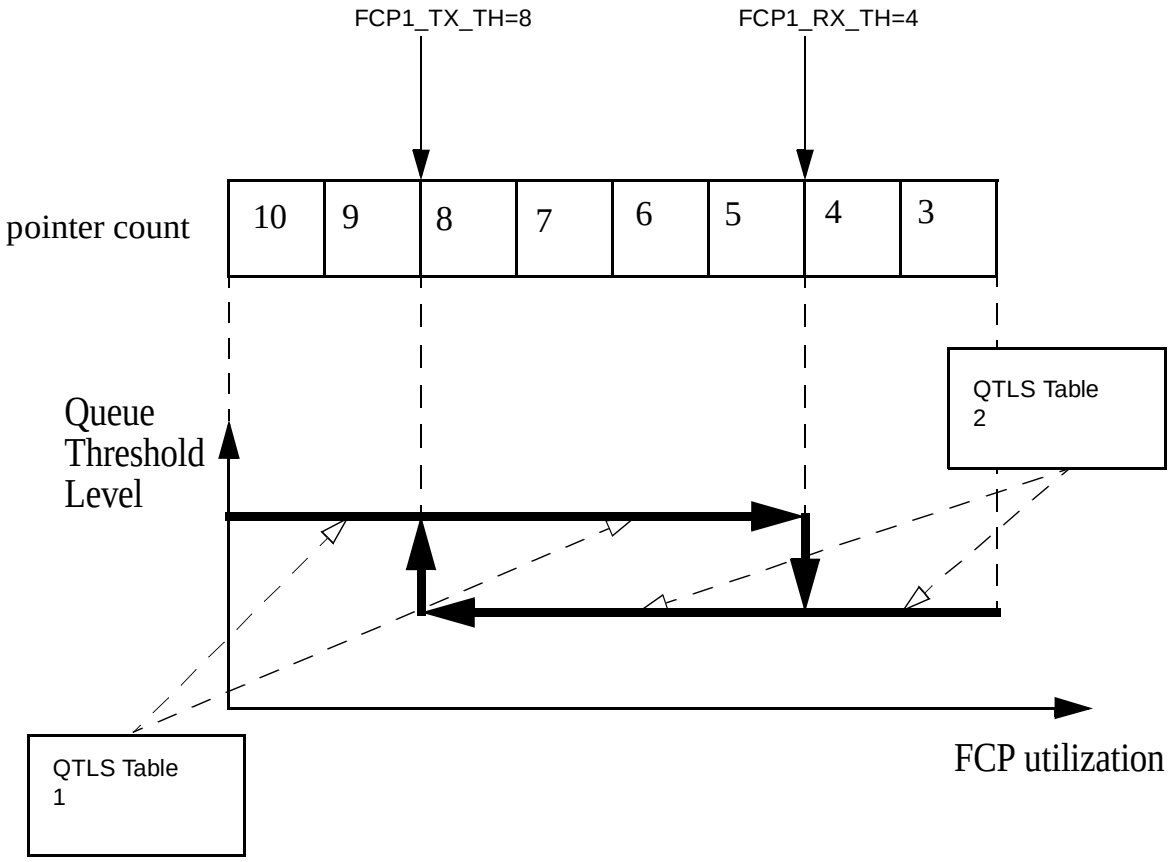


Figure 39-19. Implementing a Hysteresis Mechanism for FCP Thresholds

Note that thresholds can also be modified on the fly, to implement more complex Queue management algorithms.

39.8.3 Queue Management for GFR Service

The GFR service requires a different queue management scheme in order to be effective.

The MSP supports controllable service level for GFR eligibility.

The purpose of the GFR queueing management is to guarantee the service to eligible frames while maintaining best effort service for ineligible ones.

To be consistent with the GFR service, the CLP1 and CLP0 thresholds are being checked only for the first cell of an incoming new frame.

When a start of frame cell is being received its CLP bit is used to determine whether the current frame is eligible for service. As a rule, if the CLP bit is set the frame is considered ineligible by the F-GCRA algorithm. If the option is set to tagging and not to discard mode the cell would still be processes to the Queue management and there the CLP1 threshold would be examined.

Note that because the GFR policy is enabled, the CLP bit is identical for all of the cells in a given frame.

When a start of frame and CLP0 cell is being received, the EPD_CLP0 threshold is used to determine whether to enqueue the received frame or discard it.

For a typical implementation which includes GFR service the EPD_CLP0 threshold should be set above the EPD_CLP1 threshold (as depicted in the figure below).

The EFCI threshold is used by the transmitting side and its functionality remains unchanged.

The CLP0+1 threshold behavior remains the same.

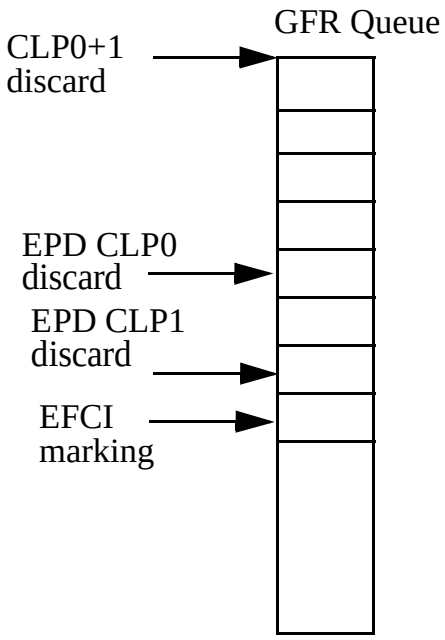


Figure 39-20. Typical Threshold Setting for GFR

39.9 Connection Tables

The receive and transmit connection tables, store host-initialized connection parameters after connection set-up and temporary parameters used during SAR and switch operation. There are a number of different formats for the connection parameters, depending on the functionality of the connection associated with the channel code in question. In some cases there is an extension to the connection table entry. Each connection table entry resides in a 32-byte space. An ATM channel is considered internal if its tables are in an internal multi-user-port RAM; it is considered external if the tables are in external memory.

Figure 39-8 lists the type of RCT, TCT, and TCTE for switched cells. The format of those structures for SARed cells, are specified in Chapter 32, "ATM Controller AAL0, AAL1, and AAL5." Note that the RISC tests the bits mentioned in the table to determine which format is applied to a given RCT/TCT.

Table 39-8. Receive and Transmit Connection Tables

ATM Switch Functionality RCT[AAL]=110 TCT[AAL]=110	Bits to select Function Receive Transmit		Receive no leaky Bucket	Receive Leaky Bucket	Transmit no UDC	Transmit UDC
Switch Unicast APC Shaping	RCTM=00	TCTM=00	**SRCT** UPCM=00	**SRCT + UPCT** UPCM!=00	**STCT** FPSMR[TUDC]=0	**STCT + STCTE** FPSMR[TUDC]=1 or ATT=01,10
Switch Unicast FIFO Shaping	RCTM=10	TCTM=10	**SRCT** UPCM=00	**SRCT + UPCT** UPCM!=00	**FTCT** FPSMR[TUDC]=0	**FTCT** FPSMR[TUDC]=1
Switch Unicast Root of Hierarchy	RCTM=01	TCTM=01	**SRCT** UPCM=00	**SRCT + UPCT** UPCM!=00	**HTCT + FTCT** FPSMR[TUDC]=0	**HTCT + FTCT (+ STCTE)** FPSMR[TUDC]=1 (or ATT=01,10)
Switch Multicast	RCTM=11	TCTM=11	**MRCT** UPCM=00	**MRCT + UPCT** UPCM!=00	**MTCT** FPSMR[TUDC]=0)	**MTCT + STCTE** FPSMR[TUDC]=1 or ATT=01,10

NOTE

To improve performance, store parameters for fast channels in internal multi-user RAM and parameters for slower channels in external memory. Channel codes 0-255 are parameters from internal memory. Connection tables for external channels are read and written from external memory when the RISC processes a cell.

NOTE

In all connection tables, fields which are not used should be initialized to zero.

39.9.1 ATM Channel Code

Each ATM channel is associated with a channel code, 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 and can't be used by the user. The channel code is used to specify a VCC when sending a TRANSMIT command, initiating the internal, external CAM or address compression tables, and when the RISC sends an interrupt to an interrupt queue.

EXAMPLE

In this example the receive structures are called RCT, and the transmit are called TCT.

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 VC * 64 bytes

(RCT+TCT) = 4 K). Channels 0–1 are reserved. The rest (1024 - 62 = 962) are external channels associated with channel codes 256–1217.

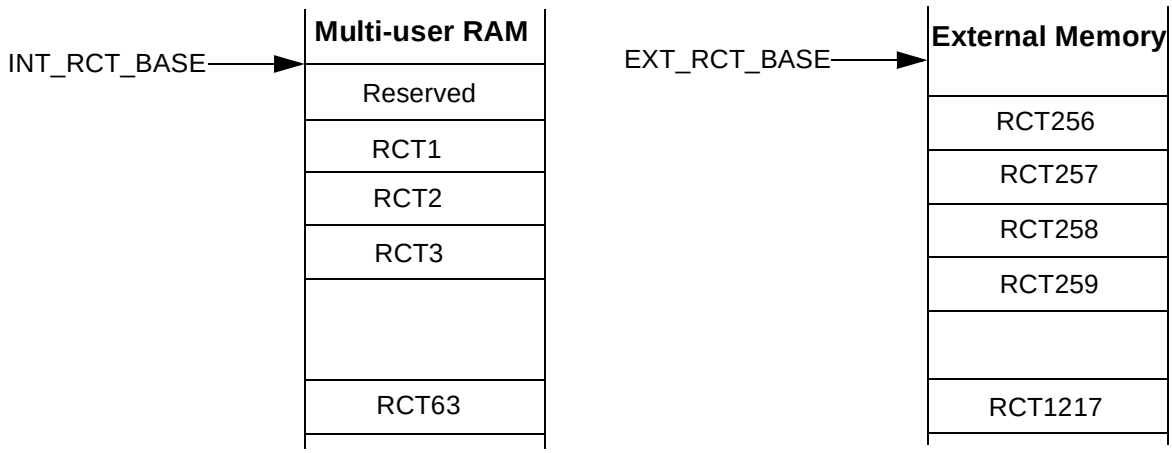


Figure 39-21. Example of a 1024-Entry Receive Connection Table

The real starting address of a RCT entry associated with channel code 3 is as follows:

INT_RCT_BASE+ (3 * 32) = INT_RCT_BASE + 96

The real starting address of a RCT entry which is associated with channel code 256 is:

EXT_RCT_BASE+ (256 * 32) = EXT_RCT_BASE + 8192

The same address calculation is used for TCT and TCTE. The TCT base address is INT_TCT_BASE, EXT_TCT_BASE. The TCTE base address is INT_TCTE_BASE, EXT_TCTE_BASE.

39.9.2 SRCT - Switched Cells Receive Parameter Table

AAL=110, RCTM=00,01,10

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00			GBL	BO		CETM	DTB	BDB	VPC	AVCON	EPDE	PPDE	UPCM		INTQ	
Offset + 0x02	RCTM		FCP	—	—	—	—	—	—	QLTS					AAL	
Offset + 0x04	FDTP(16) / EFDTP (32)/ QPTRL(32)															
Offset + 0x06	WFTMINiP(16)															
Offset + 0x08	HTCT_TCC															
Offset + 0x0A	HTCT_TAPTR/GCRA Scheduler PHY Parameter Table ptr															
Offset + 0x0C	QDCC															
Offset + 0x0E	CLP1 Count															
Offset + 0x10	CLP0 Count															
Offset + 0x12																
Offset + 0x14	GFRT															
Offset + 0x16	GFRL															
Offset + 0x18	UPCPTR															
Offset + 0x1A	TCC															
Offset + 0x1C	TAPTR/GCRA Scheduler PHY Parameter Table ptr/FSDEF															
Offset + 0x1E	MS	NCCLPIE	PMT						SEGT	ENDT	GCRA_AVCON	—	QTHIE	QDCCIE	CLPIE	PME

Figure 39-22. SRCT

For an active channel, the RISC uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes. The user can alter the interrupt enable bits on the fly.

Table 39-9 describes SRCT fields.

Table 39-9. SRCT Field Descriptions

Offset	Bits	Name	Description
0x00	0	—	Reserved. Set to zero.
	1	—	Reserved. Initialize to zero.
	2	GBL	Global. Asserting GBL signal on memory bus, 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 19.1.8.2, “Serial DMA Mode Register (SDMR).”
	3–4	BO	Byte ordering—used for data buffers. 00, 01, 11 Reserved 10 Big endian

Table 39-9. SRCT Field Descriptions (continued)

Offset	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 block, for debug purposes.
	6	**DTB**	Data buffers bus 0 Cell buffers reside on the coherent system bus. 1 Cell buffers reside on the secondary bus.
	7	**BDB**	Interrupt queues and UPCT bus 0 reside on the coherent system bus. 1 reside on the secondary bus.
	8	**VPC**	Virtual Path Connection (used for OAM termination) 0 This channel is not a VP connection 1 This channel is a VP connection
	9	**AVCON**	Auto VC On. Automatic Scheduling of channel in the APC. Valid only for STCT and HTCT MSP modes. This bit is used when the corresponding transmitter works in auto VC off mode (TCT[AVCF]=1). If set, the receiver automatically schedules the frame in the APC scheduler for transmission. 0 No Automatic rescheduling. 1 Automatic rescheduling.
	10	**EPDE**	Early Packet Discard (EPD) Enable. In this mode first cell of each frame is checked for conformance by the policer (GCRA) and queue management. If conformed, all cells in the frame will be transmitted. Leaky buckets are still being updated with no affect on discard decision. If first was discarded so will the rest of the frame. 0 EPD is disabled. 1 EPD is enabled. Setting this bit implies frame based connection (AAL5). For EPD/PPD combined mode both EPDE and PPDE bits should be set.
	11	**PPDE**	Partial Packet Discard (PPD) Enable. When a single cell is dropped by the policer, or by the queue management then all other cells associated with the current packet are dropped. The last cell in a packet will always be accepted. Only if the first cell is discarded then the Last cell will be discarded as well. Note: This bit should be reset to 0 when using GFR mode. Setting this bit implies frame based connection (AAL5). 0 PPD is disabled. 1 PPD is enabled. For EPD/PPD combined mode both EPDE and PPDE bits should be set.
	12-13	**UPCM**	UPC (Policing) Mode (See Section 39.11, “UPC Policer” for details) 00 UPC is disabled. 01 UPC is enabled, no GFR Mode 10 UPC is enabled in GFR Mode. Drop cells if CLP bit in frame changes from 1 to 0 11 UPC is enabled in GFR Mode. Set CLP bit in cells if CLP bit in frame changes from 1 to 0. Note:1. In GFR mode, if CLP bit changes from 0 to 1 cells are always dropped. See ATM forum, TM 4.1, Section 4.5.5.1 and other related sections. 2. For UPCM=1x (GFR mode), EPDE/PPDE should be cleared.
	14–15	**INTQ**	Points to one of four interrupt queues available.

Table 39-9. SRCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-1	**RCTM**	RCT Mode 00 This RCT is used for a switched channel shaped by the APC. 32 bit QPTRL is placed in address 0x4 of RCT. 01 This RCT is used for a switched channel shaped by external FIFOs. 32 bit EFDTP FIFO Descriptor Pointer to external memory is placed in address 0x4 of RCT. 10 This RCT is used for a switched channel shaped by internal FIFOs. 16 bit FDTP FIFO Descriptor Pointer to internal Multi-user RAM is placed in address 0x4 of RCT. 11 This RCT is an MRCT and is used for Multicast shaping. See Section 39.14, “Multicast.”
	2	**FCP**	Free Cell Pool 0 Use Free Cell Pool 1 1 Use Free Cell Pool 2
	3	**—**	Reserved. Set to zero.
	4	**—**	Reserved. Set to zero.
	5	**—**	Reserved. Set to zero.
	6	**—**	Reserved. Set to zero.
	7	**—**	Reserved. Set to zero.
	8	**—**	Set to zero.
	9-12	**QLTS**	Queue Level Threshold Set. This entry picks one of sixteen threshold sets for the queue. For FIFO and Hierarchical modes this entry is not valid. Instead, in the FDT (Fifo descriptor table) the entry FIFO_MODES has the QLTS entry which point to the thresholds set.
	13-15	**AAL**	AAL type: 000 AAL0 Reassembly with no adaptation layer. 001 AAL1 ATM adaptation layer 1 protocol. 010 AAL5 ATM adaptation layer 5 protocol. 011 AAL3. 100 AAl2. 101 CES. 110 MSP-Multi ServIce Protocol - switched ATM cells.
0x04	32 16	**EFDTP/QPTRL** **FDTP**	(RCT[RCTM]=01). In this mode the entry is initialized to point to the appropriate descriptor. EFDP: External FIFO descriptor Pointer (32 bits). Points to a queue descriptor in external memory. This is used to bundle multiple connections to one FIFO queue (RCT[RCTM]=10), in hierarchical mode. QPTRL: Queue Pointer Last (32 bits): Points to a buffer where the last cell received on this VCC resides. This is used for per VCC queueing (RCT[RCTM]=0). In this mode the entry is updated by the RISC. FDP: FIFO Descriptor Pointer (16 bits): Points to a queue descriptor residing in internal memory. This is used to bundle multiple connections to one FIFO queue
0x06	16	**WFTMINiP**	FOR FIFO MODE ONLY (2’nd half word): This is a pointer to the WFTMINi which is associated with the Transmitter phy FIFO Descriptor Pointer Table. See Table 39-35
0x8	—	**HTCT TCC**	Valid only for HTCT with AVCON mode enabled. Transmit Channel Code. This entry is used to schedule the cell for transmit, if the channel is not currently active. It provides the receiver with the transmitter channel number to which the received cell is forwarded.

Table 39-9. SRCT Field Descriptions (continued)

Offset	Bits	Name	Description
0xA	0-15	**HTCT TAPTR/**	Valid only for HTCT with AVCON mode enabled. **Traditional APC mode:** TAPTR: Transmitter APC Pointer. This entry is the pointer to the APC priority Table entry in multi-user RAM. This is used by the RISC to automatically add a connection to the APC (if it was previously removed since its 8 FIFOs were empty). The user must allocate place for the ATM_AVCON_BASE table, since this HTCT AVCON mode for traditional APC, uses this table to indicate if the HTCT channel is active or not. Also the user must clear the VCON bit in the HTCT table in this mode.
	0-13	**GCRA Scheduler PHY Parameter Table ptr**	**DRR APC mode:** 14 most significant bits of the pointer to the GCRA Scheduler PHY Parameter Table.
	14-15	**GCRA Scheduler PRI**	GCRA Scheduler Priority.
0x0C	—	**QDCC**	Queue Dropped Cell Count. Cells dropped due congestion in the queue.
0x0E	—	**CLP1 Count**	CLP1 cells Count.
0x10	—	**CLP0 Count**	CLP0 cells Count.
0x14	—	**GFRT**	Threshold for Maximum AAL5 packet size (MFS) for GFR. GFRT should be programed to (MFS - 1)
0x16	—	**GFRL**	Current length of AAL5 packet for GFR policing. Should be initialized to 0.
0x18	—	**UPCPTR**	Index Pointer to external UPC Parameter Table (UPCT). Aligned to 32 byte addresses. See Section 39.11, “UPC Policer” for details.
0x1A	—	**TCC**	Transmit Channel Code. This entry is used to schedule the cell for transmit, if the channel is not currently active. It provides the receiver with the transmitter channel number to which the received cell is forwarded. For RCT[RCTM]=0, TCC means STCT Channel Code, and for RCT[RCTM]=01 and for RCT[RCTM]=10 modes, TCC means FTCT Channel Code. For RCT[RCTM]=0, this information provides the receiver the pointer to the TCT info of the transmitter channel number to which the received cell is forwarded.

Table 39-9. SRCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0-15	**TAPTR/**	Valid only for RCT[RCTM]=0 and AVCON mode is enabled. **Traditional APC mode:** TAPTR: Transmitter APC Pointer. This entry is the pointer to the APC priority Table entry in multi-user RAM. This is used by the RISC to automatically add a connection to the APC (if it was previously removed since its queue was empty). **GCRA Scheduler mode:** 14 most significant bits of the pointer to the GCRA Scheduler PHY Parameter Table.
	0-13	**GCRA Scheduler PHY Parameter Table ptr**	GCRA Scheduler Priority.
	14-15	**GCRA Scheduler PRI**	For RCT[RCTM]=01 and for RCT[RCTM]=10 modes, this entry is defined as: FSDEF: See section Section 39.9.2.1, “FSDEF (FIFO Status Definition)” for bit definition.
		FSDEF	
0x1E	0	**MS**	Match Status. Valid only for CC in UDH mode. 0 Match was found. 1 Match was not found. Cell will be discarded without updating the SRCT in external memory. The CPU can use this mechanism when performing dynamic changes in the SRCT/MRCT, without the interfering of the QUICC Engine block.
	1	**NCCLPIE**	Non-Conforming CLP1 or CLP0 counters overflow interrupt enable.
	2-7	**PMT**	Performance Monitoring Table Index. One of 64 performance monitoring parameter tables is chosen fir this channel.
	8	**SEGT**	Segment OAM termination (used for OAM termination) 0 Do not terminate segment OAM cells. 1 Terminate segment OAM cells (OAM cells are sent to Raw Cell Queue).
	9	**ENDT**	End-to-end OAM termination (used for OAM termination) 0 Do not terminate end-to-end OAM cells 1 Terminate end-to-end OAM cells (OAM cells are sent to Raw Cell Queue).
	10	**GCRA Sched_AVCON**	GCRA Scheduler AVCON. Valid only if AVCON bit is set. Indicates which type of APC is used to activate the channel. 0 - Traditional APC. 1 - GCRA Scheduler
	11	**—**	
	12	**QTHIE**	Any queue threshold violation Interrupt enable not including EFCI marking.
	13	**QDCCIE**	QDCC Counter overflow Interrupt Enable
	14	**CLPIE**	CLP0C Counter or CLP1C Counter overflow Interrupt Enable
	15	**PME**	Performance monitoring Enable 0 No performance monitoring on this channel 1 Performance monitoring is enabled on this channel

39.9.2.1 FSDEF (FIFO Status Definition)

The format of the FSDEF entry in SRCT is as follows. This entry is

MODE	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hiera.	FIFOID			FSOff												
FIFO	FIFOID			-												

Figure 39-23. FSDEF bits

Table 39-10 describes the bits if the FSDEF entry.

Table 39-10. FSDEF Field Descriptions

Bits	Name	Description
0–2	**FIFOID**	Valid for Hierarchical and FIFO mode. This is the ID (0-7) of the FIFO pointed by the FDT/EFDT. This number is used by the RISC to turn on the correct bit in the FS (FIFO State) in the Multi-user RAM. User should program this entry to point to which of the 8 FIFO's to insert the incoming cell.
3-15	**FSOff**	FIFO Status Offset- For Hierarchical mode this is the Offset to FIFO status (FS) structure from FS_BASE in the multi-user RAM. The FIFO status has one bit set for each FIFO that has a non-empty queue. The user program this entry to point to the selected 8 FIFO structure.

39.9.3 Multicast Cells MRCT

AAL=110, TCTM=11

See Section 39.14, "Multicast."

39.9.4 STCT - Switched Cells Non Hierarchical Scheduling

AAL=110, TCTM=00

The STCT is used to transmit cells that are scheduled by the APC.

Figure 39-24 shows the general layout for the STCT cell that is scheduled.

	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		—	VCON	INTQ	
Offset + 0x02	TCTM		FCP	—	—	—	—	—	—	QLTS				AAL		
Offset + 0x04	QPTRF															
Offset + 0x06																
Offset + 0x08	QCC Queue Cell Count															
Offset + 0x0A	Reserved															
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															

Figure 39-24. STCT

Offset + 0x10 Offset + 0x12	CLP0 Count							
Offset + 0x14	CLP1 Count							
Offset + 0x16	APCLC							
Offset + 0x18	TVPI					TVCI		
Offset + 0x1A	TVCI				Res	CVP	CVC	CGFC
Offset + 0x1C	—	PMT		—				
Offset + 0x1E	INS_CELL_Flag	—	—		efcie	STPT	CLPIE	PME

Figure 39-24. STCT

The RISC updates only entries 0x00-0x17 (24 bytes), and 1 byte located in 0x1e; other bytes are read by the RISC but are not updated by it. The CPU controlling the connection can change relevant bits from entries 0x18-0x1F (not including 0x1e) on the fly, and the RISC will act upon the new value, next time the TCT is used. The CPU shouldn't change the content of the byte located in address 0x1e, but can only read the status of this byte.

Table 39-11 describes general STCT fields.

Table 39-11. STCT Field Descriptions

Offset	Bits	Name	Description
0x0	0-1	—	Reserved. Initialize to zero.
	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 19.1.8.2, "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 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.
	6	**DTB**	Data buffers bus. Cell location (where the FCP resides) If Insert cell mode is enabled and the cell resides in external memory, this bit should reflect the bus location of the cell to be transmitted. 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the secondary bus.
	7	**BDB**	Interrupt queues bus. If Insert cell mode is enabled and the CC>255 this bit should reflect the bus location of the Insert Cell pointers Table see Section 39.15.3, "Insert Cell Mode." 0 Resides on the coherent system bus. 1 Resides on the secondary bus.

Table 39-11. STCT Field Descriptions (continued)

Offset	Bits	Name	Description
	8	**AVCF**	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more cells are queued. The number of cells in the VC queue is specified in the TCT[QCC] field. 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. This mode is used for root nodes in hierarchical APC. Root nodes cannot be removed from the scheduler, since there is no mechanism to determine if there are cells scheduled in the leaves beneath the root (the assumption is that there are always cells to be scheduled under the root nodes which schedule to VPs). 1 The APC removes this VC from the schedule table, if TCT[QCC]=0, the VC queue has been empty for one cell interval time for this connection. When the APC remove this VC it resets TCT[VCON] (VCON=0). The VC is automatically activated by the receiver, if a cell arrives, setting TCT[VCON] (VCON=1). See Table 39-13 in the explanation of HCTC[AVCF]
	9	—	Reserved. Set to zero.
	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. 11 Reserved.
	12	—	Reserved. Set to zero.
0x00	13	**VCON**	Virtual channel is on. In non AVCON/AVCOFF Mode: Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPT] (stop transmit), the RISC 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 RISC clears VCON. In AVCON/AVCOFF Mode: Should be cleared by the host. If TCT[QCC]=0 and TCT[AVCOFF] bit is set the APC is automatically removes this VC from the APC slots and resets TCT[VCON] bit, (VCON=0) When cell arrives on the Receiver side and RCT[AVCON] bit is set The VC is automatically activated by entering the VC to an APC slot and sets TCT[VCON] bit, (VCON=1).
	14-15	**INTQ**	Points to one of four interrupt queues.

Table 39-11. STCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-1	**TCTM**	TCT Mode 00 This is a STCT used as a leaf node for APC shaping 01 This is a HTCT used for a root node of hierarchical shaping 10 This is a FTCT used for Fast FIFO shaping 11 This is a MTCT used for Multicast shaping
	2	**FCP**	Free Cell Pool 0 Use Free Cell Pool 1 1 Use Free Cell Pool 2
	3-8	—	Reserved. Initialize to zero
	9-12 .	**QLTS**	Queue Level Threshold Set. This entry picks one of sixteen threshold sets for the queue
	13-15	**AAL**	AAL type: 000 AAL0 Reassembly with no adaptation layer. 001 AAL1 ATM adaptation layer 1 protocol. 010 AAL5 ATM adaptation layer 5 protocol. 011 AAL3. 100 AAl2. 101 CES. 110 MSP-Multi ServIce Protocol - switched ATM cells.
0x04		**QPTRF**	Queue Pointer First (32 bits): Points to a buffer where the first cell to be transmitted on this queue resides.
0x08	0-15	**QCC**	Queue Cell Count: The number of cell buffers currently queued for this channel. Used by RISC. If QCC=0 there are no cells in the queue. If TCT[VCON]=0 the channel has been deactivated by the RISC. Initialize to zero.
0x0A	0-15	**Reserved**	Reserved. Initialize to zero.
0x0C	0–7	**Rate Remainder**	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Initialize to 0.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract.
0x10	—	**CLP0 Count**	CLP0 cell count. Initialize to 0.
0x14	—	**CLP1 Count**	CLP1 cell count. Initialize to 0.
0x16	—	**APCLC**	APC Linked Channel. Used by RISC. Initialize to zero.
0x18	0-11	**TVPI**	Transmit VPI. Replaces VPI of cell if CVP bit is set (including GFC bits if CGFC is set)
	12-15	**TVCI**	Most significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set

Table 39-11. STCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1A	0-11	**TVCI**	Least significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set
	12	**Reserved**	Reserved. Initialize to zero.
	13-15	**CVP/CVC/CGFC**	CVP/CVC/CGFC (address translation, part of ATM header) 000 no translation 001 NOT valid 010 Change VCI Portion of ATM address 011 NOT valid 100 Change VPI Portion of ATM address (including GFC) 101 Change VPI Portion of ATM address (NOT including GFC) 110 Change VCI & VPI Portion of ATM address (including GFC) 111 Change VCI & VPI Portion of ATM address (NOT including GFC)
0x1C	0-1	—	Reserved. Set to zero
	2-7	**PMT**	Performance monitoring Table Index. One of 64 performance monitoring parameter tables is chosen for this channel
	8-15	—	Reserved. Set to zero
0x1E	0	**INS_CELL_Flag**	Insert Cell Flag. Initialized to 0. Set by the RISC when insert cell host command is performed by the user, for inserting a cell to this connection. See Section 39.20.2, “MSP Insert Cell Command“. Reset by the RISC on completion of the inserted cell operation. See 39.15.3, “Insert Cell Mode”. The CPU can only read this status bit in order to verify that the inserted cell was actually transmitted for this channel. Only after this bit was cleared another insert cell host command for this channel could be issued.
	1-11	—	Reserved. Set to zero
	12	**efcie**	EFCI threshold interrupt enable.
	13	**STPT**	Stop transmit. Initialize to 0. When the host sets this bit, the RISC deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. The RISC reads 32 bytes of TCT, but writes back only 24 bytes. Therefore this bit is not altered by the RISC.
	14	**CLPIE**	CLP0C Counter or CLP1C Counter overflow Interrupt Enable
	15	**PME**	Performance monitoring Enable 0 No performance monitoring on this channel 1 Performance monitoring is enabled on this channel

39.9.5 FTCT - FIFO TCT

TCTM=10

The FTCT is used to transmit switched cells that are scheduled by the FIFOs.

Figure 39-25 shows the general layout for the FTCT.

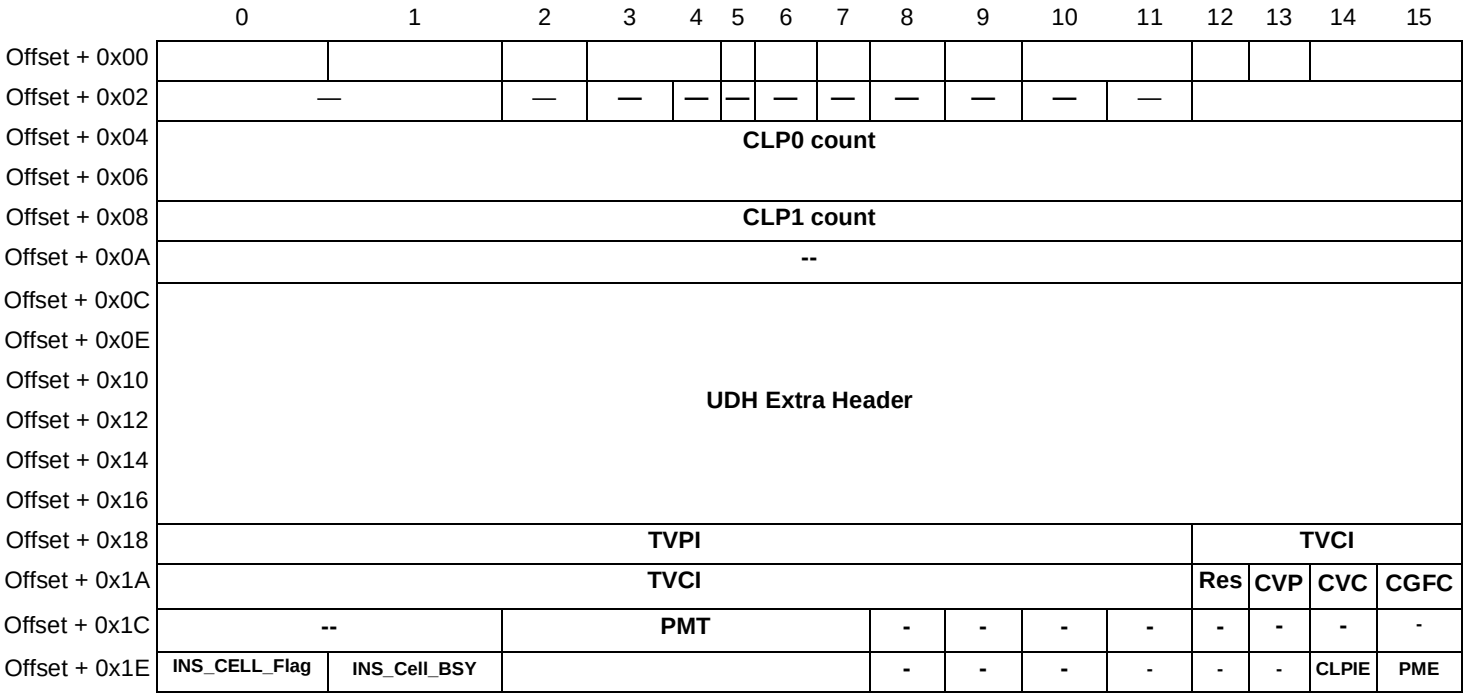


Figure 39-25. FTCT

The RISC updates only entries 0x00-0xB (12 bytes) and 1 byte located in 0x1e; other bytes are read by the RISC but are not updated by it. The CPU controlling the connection can change relevant bits from entries 0xC-0x1F (not including 0x1e) on the fly, and the RISC will act upon the new value, next time the TCT is used. The CPU shouldn't change the content of the byte located in address 0x1e, but can only read the status of this byte.

Table 39-12 describes general FTCT fields.

Table 39-12. FTCT Field Descriptions

Offset	Bits	Name	Description
0x00	0-15		Reserved. Set to zero.
0x02	0-15		Reserved. Set to zero.
0x04		**CLP0 Count**	CLP0 cell count. Initialize to 0.
0x08	—	**CLP1 Count**	CLP1 cell count. Initialize to 0.
0x0A			Reserved. Initialize to zero
0x0C-0x17	-	**UDH**	User defined header (extra header) to be transmitted before cell.
0x18	0-11	**TVPI**	Transmit VPI. Replaces VPI of cell if CVP bit is set (including GFC bits if CGFC is set)
	12-15	**TVCI**	Most significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set

Table 39-12. FTCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x01A	0-11	**TVCI**	Least significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set
	12	**Reserved**	Reserved. Initialize to zero.
	13-15	**CVP/CVC/CGFC**	CVP/CVC/CGFC (address translation) 000 no translation 001 NOT valid 010 Change VCI Portion of ATM address 011 NOT valid 100 Change VPI Portion of ATM address (including GFC) 101 Change VPI Portion of ATM address (NOT including GFC) 110 Change VCI & VPI Portion of ATM address (including GFC) 111 Change VCI & VPI Portion of ATM address (NOT including GFC)
0x1C	0-1	—	Reserved. Set to zero
	2-7	**PMT**	Performance monitoring Table Index. One of 64 performance monitoring parameter tables is chosen for this channel
	8-15	—	Reserved. Set to zero
0x1E	0	**INS_CELL_Flag**	Insert Cell Flag. Initialized to 0. Set by the RISC when insert cell host command is performed by the user, for inserting a cell to this connection. See Section 39.20.2, "MSP Insert Cell Command". Reset by the RISC on completion of the inserted cell operation. See 39.15.3, "Insert Cell Mode". The CPU can only read this status bit in order to verify that the inserted cell was actually transmitted for this channel. Only after this bit was cleared another insert cell host command for this channel could be issued.
	1	**INS_CELL_BSY**	Insert Cell Busy flag. Set by the RISC after Insert Cell host command for FTCT/HTCT modes (see Section 39.20.2, "MSP Insert Cell Command") was issued for this FTCT channel, and the FCP was in busy condition. When set means that the insert cell host command will not insert the requested cell due to the lack of FCP resources. When it is set the CPU should allocate more memory for the FCP, and after that to issue again the insert cell host command for this channel.
	2-13	**-**	res
	14	**CLPIE**	CLP0C Counter or CLP1C Counter overflow Interrupt Enable
	15	**PME**	Performance monitoring Enable 0 No performance monitoring on this channel 1 Performance monitoring is enabled on this channel

39.9.6 HTCT–Root of Hierarchy Parameters

TCTM=01

This TCT is used as a root of a hierarchical branch (bit HTCT=1), in Switch mode, and in non multicast mode. No actual cell is transmitted with this TCT, rather a FIFO structure which is pointed by an external Fifo descriptor (EFDT) is accessed from external memory and a cell is sent from there.

Figure 39-26 shows the general layout for the HTCT cell that is scheduled.

	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			VCON	INTQ	
Offset + 0x02	TCTM		FCP	-										AAL		
Offset + 0x04	--															
Offset + 0x06																
Offset + 0x08	FFS_TMP															
Offset + 0x0A	--			FSOff												
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10	--															
Offset + 0x12																
Offset + 0x14	--															
Offset + 0x16	APCLC															
Offset + 0x18	EFDT_FIRST															
Offset + 0x1A																
Offset + 0x1C	--															
Offset + 0x1E	--													STPT	-	-

Figure 39-26. HTCT

The RISC updates only entries 0x00-0x17 (24 bytes), other bytes are read by the RISC but are not updated by it. The CPU controlling the connection can change relevant bits from entries 0x18-0x1F on the fly, and the RISC will act upon the new value, next time the TCT is used.

Table 39-13 describes general HTCT fields.

Table 39-13. HTCT Field Descriptions

Offset	Bits	Name	Description
0x00	0	—	Reserved. Set to zero.
	1	—	Reserved. Set to zero.
	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 19.1.8.2, "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 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.

Table 39-13. HTCT Field Descriptions (continued)

Offset	Bits	Name	Description
	6	**DTB**	Data buffers bus. Cell location (where the FCP resides) If Insert cell mode is enabled and the cell resides in external memory, this bit should reflect the bus location of the cell to be transmitted. 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the secondary bus.
	7	**BDB**	EFDT and Interrupt queues bus. If Insert cell mode is enabled and the CC>255 this bit should reflect the bus location of the Insert Cell pointers Table see 39.15.3, “Insert Cell Mode. 0 Resides on the coherent system bus. 1 Resides on the secondary bus.
	8	**AVCF**	Auto VC off. Determines APC behavior when the last cell associated with this HTCT has been sent and no more cells are queued under this HTCT, see Table 32-19, in the Global ATM Parameter Table, ATM _AVCON_BASE. 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 and resets TCT[VCON] bit, when there is no cell to transmit under this VC. The VC is automatically activated by the receiver, if a cell arrives, and sets TCT[VCON] bit.
	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. 11 Reserved.
	12	—	Reserved. Set to zero.
0x00	13	**VCON**	Virtual channel is on. In non AVCON/AVCOFF Mode: Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPT] (stop transmit), the RISC 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 RISC clears VCON. In AVCON/AVCOFF Mode: Should be cleared by the host.
	14-15	**INTQ**	Points to one of four interrupt queues.

Table 39-13. HTCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-1	**TCTM**	TCT Mode 00 This is a STCT used as a leaf node for APC shaping 01 This is a HTCT used for a root node of hierarchical shaping 10 This is a FTCT used for Fast FIFO shaping 11 This is a MTCT used for Multicast shaping
	2	**FCP**	Free Cell Pool 0 Use Free Cell Pool 1 1 Use Free Cell Pool 2
	3-12		reserved set to 0.
	13-15	**AAL**	AAL type: 000 AAL0 Reassembly with no adaptation layer. 001 AAL1 ATM adaptation layer 1 protocol. 010 AAL5 ATM adaptation layer 5 protocol. 011 AAL3. 100 AAl2. 101 CES. 110 MSP-Multi ServIce Protocol - switched ATM cells.
0x04-0x07		—	Reserved. Set to zero.
0x08	Hword	**FFS_TMP**	initialize to zero. used by RISC only
0x0A	0-2	**Reserved**	initialize to zero.
	3-15	**FSOff**	FIFO Status Offset. Offset to FIFO status (FS) from **FS_BASE** in the multi-user RAM. The FS is initialized to zero, and is changed by the RISC. The FIFO status has one bit set for each FIFO that has a non-empty queue.
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. Initialize to 0.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract.
0x10	Word	—	Reserved. Set to zero.
0x14	Hword	—	Reserved. Set to zero.
0x16	—	**APCLC**	APC Linked Channel. Used by RISC. Initialize to zero.
0x18	Word	**EFDT_FIRST**	Pointer to First entry in External FIFO Descriptor Table.
0x1C	-	—	Reserved. Set to zero.
0x1E	0-12	—	Reserved. Set to zero.
	13	**STPT**	Stop transmit. Initialize to 0. When the host sets this bit, the RISC deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. The RISC reads 32 bytes of TCT, but writes back only 24 bytes. Therefore this bit is not altered by the RISC.
	14-15		Reserved. Set to zero.

39.9.7 MTCT—Multicast TCT

TCTM=11

See Section 39.14, “Multicast.”

39.9.8 Switched/Multicast Transmit Connection Table Extension (STCTE/MTCTE)

Figure 39-27 shows the general layout for the STCTE/MTCTE.

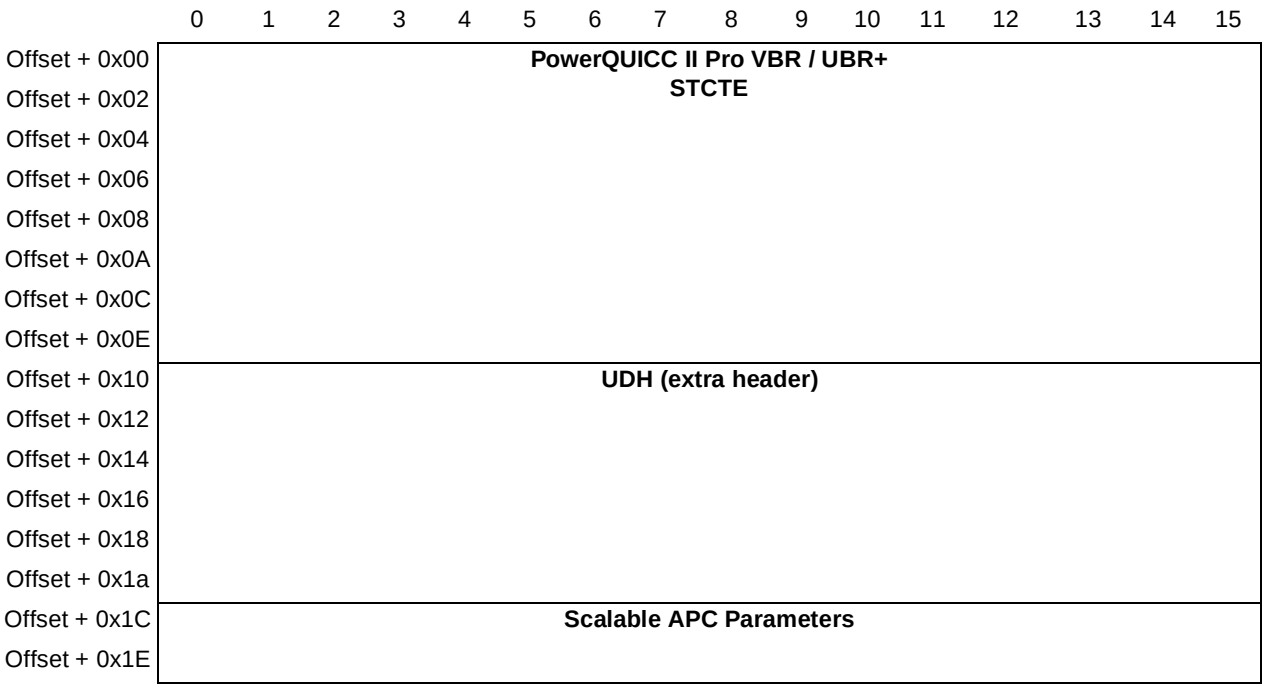


Figure 39-27. STCTE/MTCTE

Table 39-14 describes general STCTE fields.

Table 39-14. STCTE/MTCTE Field Descriptions

Offset	Bits	Name	Description
0x00-0x0E	—	**Param**	PowerQUICC II Pro param for VBR / UBR+. Valid only for APC mode (for multicast working in FIFO-MODE this entry isn't valid).
0x10-0x1B	—	**UDH-EXTRA HEADER**	Holds the channel Extra header for User Defined Cell mode (UDC)
0x1C-0x1F		**Scalable APC Parameters**	Holds the APC scalable Mode parameters

39.10 Scheduling

The PowerQUICC II ProMSP allows for multiple configurations of the scheduler. The scheduler at any point in time selects a cell to be transmitted. For multiphy systems there is a separate scheduler for each PHY on the Utopia. The modes of operations per PHY are as follows:

Eight FIFOs which can be configured as fixed priority or Weighted Fair Queuing or a mixed mode. In this configuration all ATM connections are scheduled into one of the eight FIFOs. In this mode there is no per VC queueing, and the queues contain cells from many different connections. Nevertheless billing, address translation, extra-header and PM functions are still performed on a per-connection basis.

OR

Hierarchical APC, which includes eight prioritized Power QUICC II like APCs at the top level of the hierarchy, and optionally under each slot of the APC, eight FIFOs that can be configured only in fixed priority. Connections scheduled in the APC itself, are dealt as per-VC queueing (like the MPC8260). Connections in hierarchical mode are scheduled as a bundle (normally per-VP) by the APC, and then inside the bundle, the scheduling is done by the FIFOs, according to a Fixed priority scheme. Note that the allocation of connection to different queues is implementation dependent.

Hierarchical shaping can be used in the following way: VPs are shaped at the top level of the hierarchy by allocating a slot in the APC at the frequency needed by the rate of the VP. The VCCs under the VP are scheduled in the FIFOs, allowing for bandwidth allocation for groups of VCs. A VC that does not have cells to send gives its bandwidth to other VCs in the FIFO (which means that FIFO scheduling is a work conserving mechanism). CES connections that need exact shaping, are scheduled by the APC in the top level of the hierarchy.

Figure 39-28 depicts a scheduling system with APC and (optionally) FIFO at the lower level of the hierarchy.

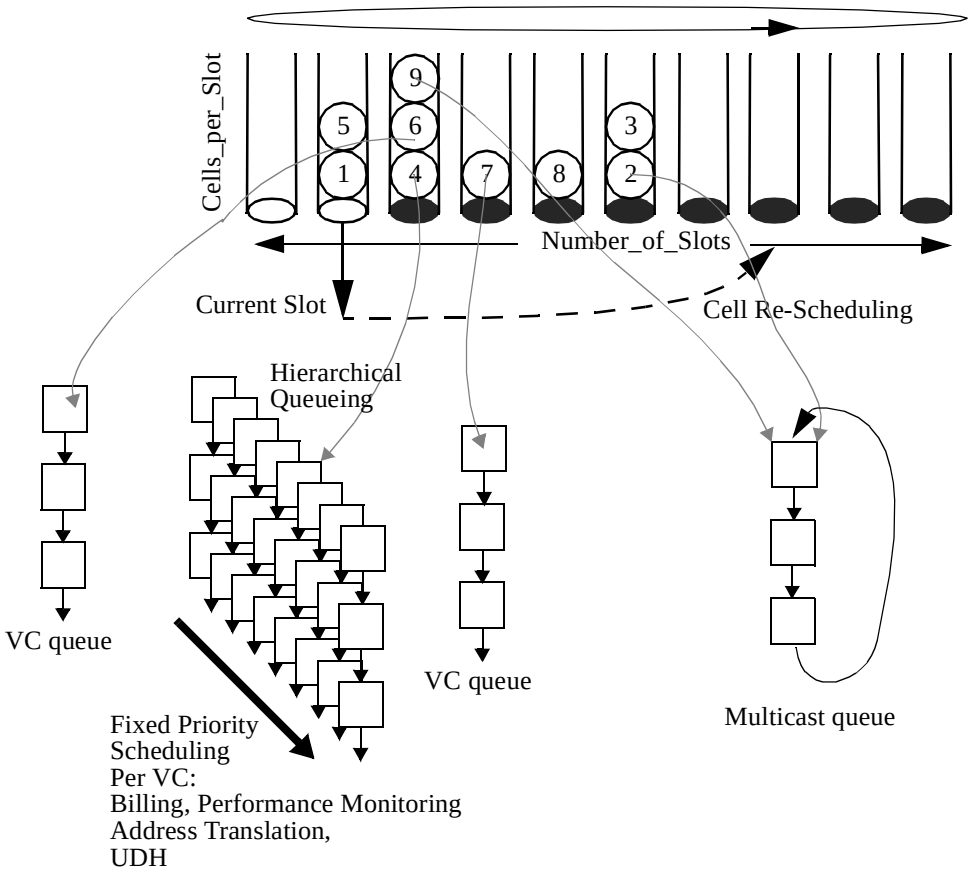


Figure 39-28. APC with Hierarchical FIFOs

Figure 39-29 depicts the WFQ scheduling mode.

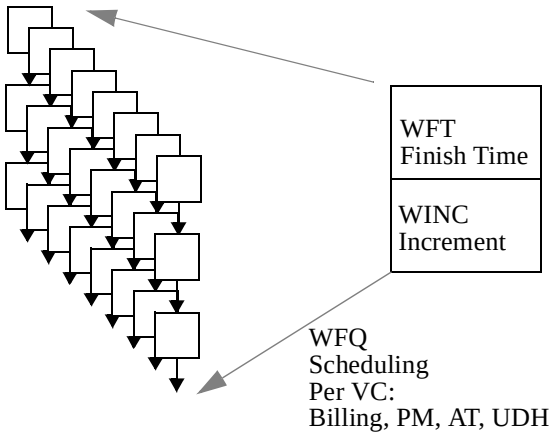


Figure 39-29. WFQ Mode

Note that the Scheduling mode is determined by the user on a per-PHY basis.

39.10.1 WFQ Weighted Fair Queueing

In weighted fair queueing, all queues are examined, and a cell from the queue with the smallest finish is transmitted. The relative data rate for the queues is determined by the WINC parameter as programmed in the Scheduler Parameter Table (see Section 39.10.3.2, “Scheduler Parameter Table in WFQ Mode (AFM=0)”). An example of programming the WINC value Is given at Section 39.10.3.3, “Examples of settings.”

39.10.2 Transmitter: Scheduler Data Structure

Each PHY has a Scheduling Parameter Table associated with it (this table is also called “APC Parameter Table,” see Section 32.3.6.1, “APC Parameter Tables”). In this table there is an entry that distinguishes between the two scheduling mechanisms (APC or eight FIFOs). APC data structure consists of three elements: Scheduler Parameter Table, the APC priority table, and the APC scheduling tables.

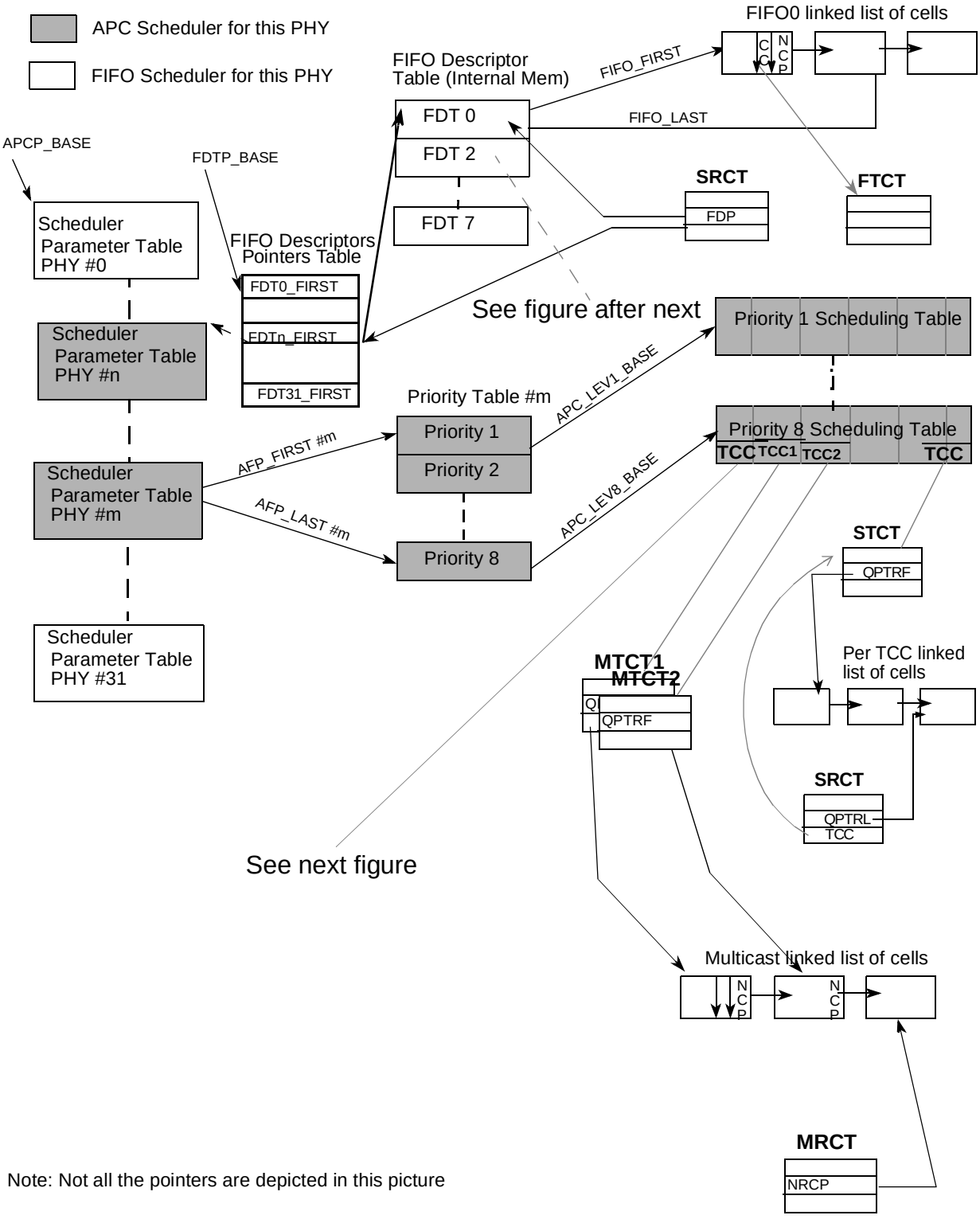


Figure 39-30. Switch Data Structures

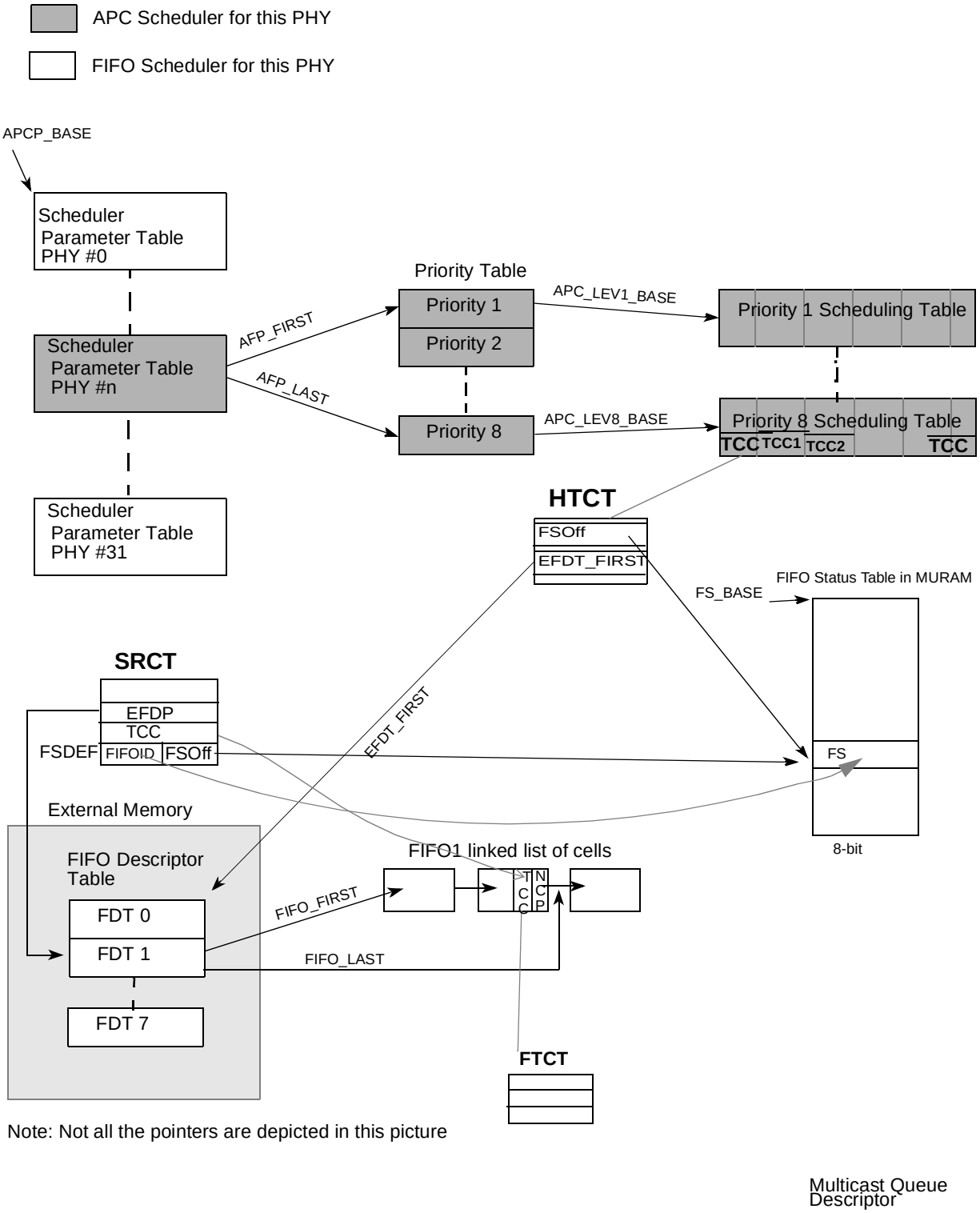
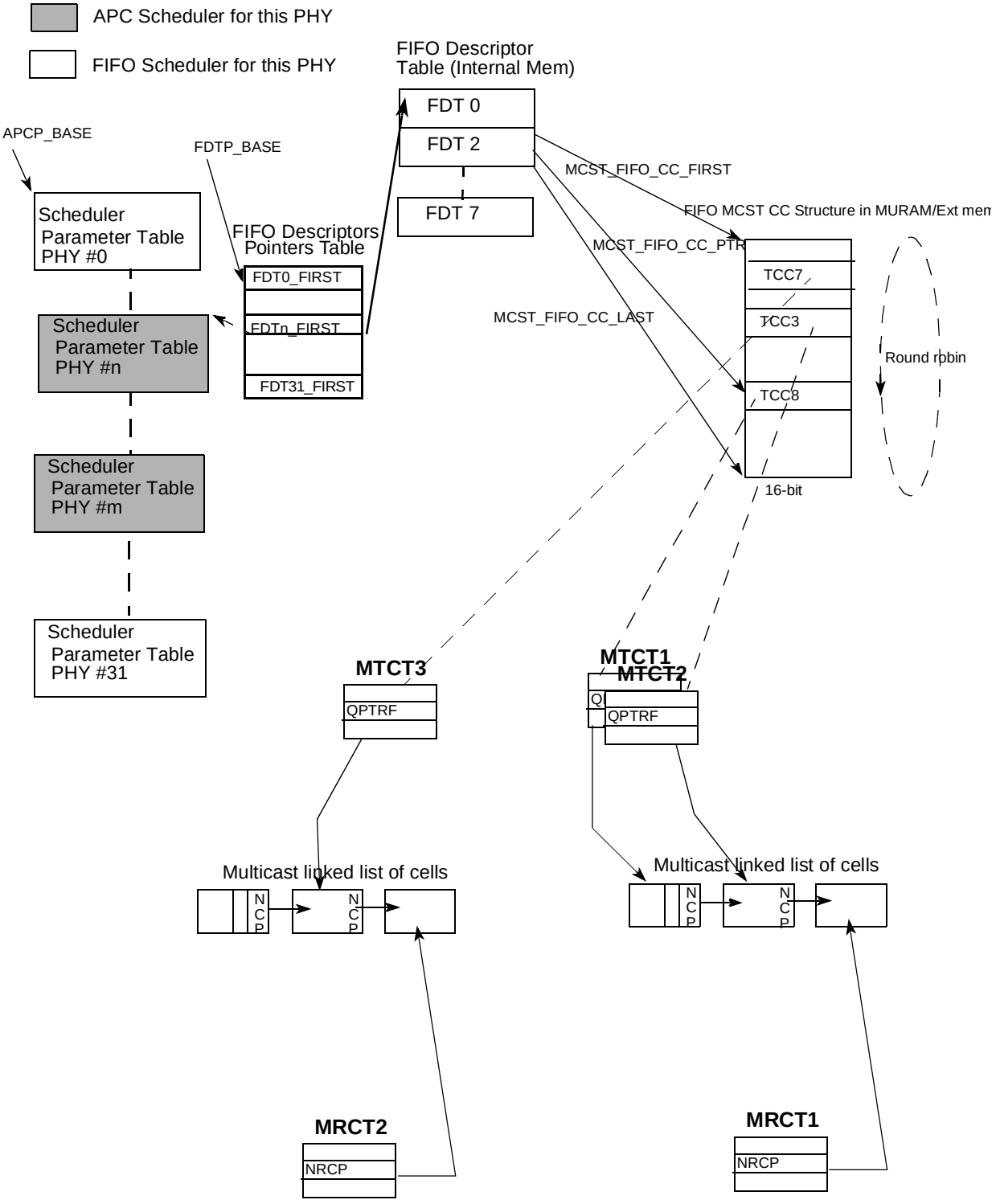


Figure 39-31. Detailed Hierarchical Switch Data Structures



Note: Not all the pointers are depicted in this picture

Figure 39-33. Multicast FIFO Data Structure

39.10.3 Scheduler Parameter Table

Each PHY's Scheduler parameter table 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 39.17, "ATM Parameter RAM Map," 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 Scheduler Parameter Table is given by APCP_BASE + PHY# *32.

Figure 39-34 shows the general layout for the Scheduler Parameter Table

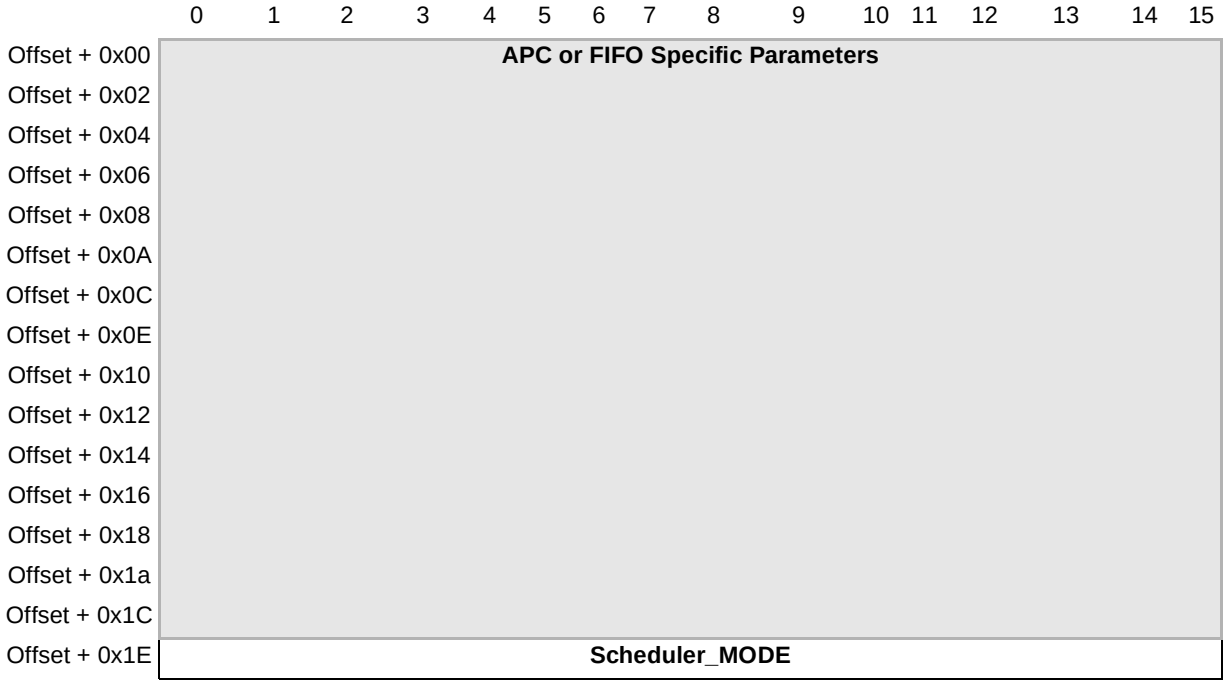


Figure 39-34. Scheduler Parameter Table

39.10.3.1 Scheduler_MODE Entry

Figure 39-35 shows the layout of the Scheduler_MODE entry in the Scheduler Parameter Table.

The functionality of this entry is determined by the setting of the AFM bit.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	**AFM**	Function Specific Bits														

Figure 39-35. Scheduler_MODE

Table 39-15. Scheduler_MODE Field Descriptions

Bits	Name	Description
0	**AFM**	Scheduler Mode (Used only if GMODE[MSP]=1) 0 This PHY scheduling is eight FIFO queues. 1 This PHY scheduling is PowerQUICC II Pro APC Like.
1-15		Function specific entries described in the following paragraphs.

39.10.3.2 Scheduler Parameter Table in WFQ Mode (AFM=0)

The following table describes the FIFO specific Parameters in the Scheduler Parameter Table when the Scheduler_MODE entry is configured as FIFO in WFQ mode - for multicast or non-multicast modes. This format is used when AFM=0.

Table 39-16. Scheduler Parameter Table

Offset[1]	Name	Width	Description
0x00-0x0F	WFTi (i=0..7)	Hword	Weighted Finish Time. Eight 16-bit FTMs, one for each FIFO. The FTM is used to determine the finish time of the next cell being transmitted from the queue. FTM at offset zero corresponds to FDT0 (FIFO number 0), FTM at offset 0x0E corresponds to FIFO7. See instructions for initialization in field description, Section 39.10.3.2.1, "WFT."
0x10-0x1D	WINCi (i=0..6)	Hword	Weighted Increment. Eight 16 bit increments, one for each FIFO. INC at offset 0x10 corresponds to FDT0 (FIFO number 0), INC at offset 0x1C corresponds to FIFO6.
0x1E	Scheduler_MODE	Hword	This entry also includes INC7. See description in Section 39.10.3.2.2, "Scheduler_MODE Entry for WFQ AFM=0. "

[1]Offset values are to APCP_BASE+PHY# × 32

39.10.3.2.1 WFT

Figure 39-36 shows the format of the Weighted Fair Queueing (WFQ) Finish Time entry in the Scheduler Parameter Table. There are eight such entries in the table, one for each queue. These entries are initialized by the CPU, and are not accessed when the switching function is enabled.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00,.. 0x0E	**FE**	**FTM**												**FID**		

Figure 39-36. WFT Entry

Table 39-17 describes the WFT fields

Table 39-17. WFT Field Descriptions

Bits	Name	Description
0	FE	FIFO Empty. Unicast mode/Multicast mode with AUTO FE Mechanism - This bit is set automatically by the RISC if the corresponding FIFO has no cells to transmit. This bit should be initialized to 1 by the CPU. Multicast mode without AUTO FE Mechanism - This bit should be initialized to 1 by the CPU. Since this bit is not changed automatically by the RISC, the user should clear this bit to activate the FIFO, only after verifying that all the related Multicast structures are properly initialized. See Section 39.14, "Multicast." The CPU also clears the FE bit before it issues a FIFO_EN event. See Section 39.14.5.2, "On-Line Modification of FIFO WFQ Multicast Mode without Auto FE Mechanism."
1-12	FTM	Finish time. This bit is automatically updated by the RISC. Initialize with WINC[INC] value.
13-15	FID	FIFO ID. The CPU initializes the number of the FIFO in these bits; the value is determined by the offset of the entry from the Scheduler Parameter Table base for this PHY: Offset Value 0000 - 000 0010 - 001 0100 - 010 0110 - 011 1000 - 100 1010 - 101 1100 - 110 1110 - 111

Figure 39-37 shows the format of the Weighted Fair Queueing (WFQ) Increment entry in the Scheduler Parameter Table. There are seven such entries in the table, one for each queue except for queue number 7. The INC value for this queue resides in the Scheduler_MODE entry. These entries are initialized by the CPU, and are not accessed when the switching function is enabled.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x10,.. 0x1E	--	INC												--		

Figure 39-37. WINC Entry

Table 39-18 describes the WINC fields.

Table 39-18. WINC Field Descriptions

Bits	Name	Description
0		Reserved. Initialize to zero.
1-12	**INC**	Weighted Increment value. This value determines the relative bit rate for the FIFO. If all FIFOs are backlogged the absolute bit rate of each FIFO is 1/INC. If some of the FIFOs do not have cells to transmit, the total bandwidth of the port is divided among the backlogged FIFOs according to their relative INC values. For strict/fixed priority among FIFOs program INC=0 for all 8 FIFOs in the PHY - FIFO 0 has the highest priority. For round robin priority among FIFOs set INC=1 for all the FIFOs. Note-When using fixed priority, with combination of Multicast FIFOs and Unicast FIFOs, it is preferred to use AUTO_FE mode for the multicast connections. This in order to prevent a condition where a multicast FIFO with FID=0, for example, which its FE isn't dynamically changed by the RISC, be always chosen by the RISC even if it is empty.
13-15		Reserved. Initialize to zero.

39.10.3.2.2 Scheduler_MODE Entry for WFQ AFM=0

Figure 39-38 shows the layout of the Scheduler_MODE entry in the Scheduler Parameter Table. This format is valid when AFM=0.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	**AFM**	**INC7**												**--**		

Figure 39-38. Scheduler_MODE

Table 39-19 describes Scheduler_MODE fields.

Table 39-19. Scheduler_MODE for WFQ Field Descriptions

Bits	Name	Description
0	**AFM=0**	Scheduler Mode 0 This PHY scheduling is eight FIFO queues
1-12	**INC7**	Weighted Increment value for FIFO number 7.
13-15		Reserved. Initialize to zero

NOTE

PHYs that do not have switched cells, and are running the MPC8260 SAR must set this bit.

39.10.3.3 Examples of settings

39.10.3.3.1 WFQ among all FIFOs

The WFQ mechanism can be used to implement many scheduling schemes of Weighted Fair Queueing among all eight FIFOs. In this mode each FIFO has a INC associated with it.

For example:

Table 39-20. Example of WFQ Among All FIFOs

	FIFO0	FIFO1	FIFO2	FIFO3	FIFO4	FIFO5	FIFO6	FIFO7
INC	3	5	2	30	2	5	30	3

The formula for getting the bandwidth for each FIFO, assuming FIFO #i has 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.

39.10.3.3.2 Strict Priorities Among all FIFOs

Strict priority is a mode in which as long a high priority FIFO has cells queued to be transmitted this FIFOs is chosen. To implement strict priority among all FIFOs (FIFO0 is highest, and FIFO7 is lowest priority) program all INC values to zero.

39.10.3.4 Mixed mode

To implement mixed mode where a number of FIFOs are strict, and others are scheduled with WFQs: The strict priority FIFOs are assigned to the low numbered FIFOs (starting from FIFO number 0 - FIFO0), these FIFOs are assigned INC=0. Other FIFOs are assigned INC according to their relative weight.

Table 39-21. Example of Mixed Mode WFQ

	FIFO0	FIFO1	FIFO2	FIFO3	FIFO4	FIFO5	FIFO6	FIFO7
INC	0	0	2	30	2	5	30	3

In this example FIFO0 and FIFO1 are strict; as long as FIFO0 has cells it is chosen to transmit. If it has no cells, FIFO1 is chosen and if it is also empty, one of FIFO2 to FIFO7 is chosen. Of the remaining bandwidth (after FIFO0 and FIFO1) the remaining FIFOs are allocated the bandwidth according to their relative weight.

39.10.4 FIFO Descriptor Table (FDT/EFDT)

39.10.4.1 FDT/EFDT for Non-Multicast FIFO

Each PHY's FIFO descriptor Table table holds pointers to the cells link list associated with this FIFO (refer to Figure 39-29). This table resides in multi-user RAM. Table 39-22 shows the structure of a FIFO descriptor table For Hierarchical connection the same data structure is used. The difference is that it reside in the external memory, therefor called External FDT (EFDT).

Table 39-22. FIFO Descriptor Table Entry (FDT/EFDT)

Offset	Name	Width	Description
0x00	FIFO_FIRST	Word	Pointer to the first Cell in FIFO
0x04	FIFO_MODES	Hword	DMA modes and bus selection. This Hword is valid for non hierarchical FIFO mode ONLY and is used for getting the cell's data and the FTCT associated with them.see Section Figure 39-39., "FIFO_QLTS Entry."
0x06		Hword	Modes of operation for this FIFO. See section Section 39.10.4.2, "FIFO_MODES Entry."
0x08	FIFO_LAST	Word	Pointer to the last cell in FIFO
0x0C	--	Hword	Reserved. Initialize to Zero.
0x0E	FIFO_COUNT	Hword	Number of cells in FIFOi (Limited to 2^15-1entries per queue)

39.10.4.2 FIFO_MODES Entry

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x04	—	—	GBL	BO		CETM	DTB	BDB	FCP					EFCIE	INTQ	
0x06	-	-	--						-	QLTS				--		

Figure 39-39. FIFO_QLTS Entry

Table 39-23. FIFO_MODES Table Entry

Offset	Bit	Name	Description
0x04	0-1	-	Reserved. Initialize to zero.
	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 19.1.8.2, "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 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.
	6	DTB	Data buffers bus. Cell location (where the FCP resides) If Insert cell mode is enabled and the cell resides in external memory, this bit should reflect the bus location of the cell to be transmitted. 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the secondary bus.
	7	BDB	Interrupt queues bus. If Insert cell mode is enabled and the CC>255 this bit should reflect the bus location of the Insert Cell pointers Table see Section 39.15.3, "Insert Cell Mode." 0 I reside on the coherent system bus. 1 reside on the secondary bus.
	8	FCP	Free Cell Pool 0 Use Free Cell Pool 1 1 Use Free Cell Pool 2
	9-12		reserved. Initialize to zero.
	13	EFCIE	EFCI indication set interrupt enable
	14-15	INTQ	Points to one of four interrupt queues.

Table 39-23. FIFO_MODES Table Entry (continued)

Offset	Bit	Name	Description
0x06	0	-	reserved, initialize to zero.
	1	-	reserved, initialize to zero.
	2-7	-	reserved, initialize to zero.
	8	-	reserved, initialize to zero.
	9-12	**QLTS**	Queue Level Thresholds Set. One of 16 Threshold Sets is chosen. From sets located at QLTS_BASE
	13-15	**Reserved**	Reserved. Set to zero.

39.10.4.3 FDT for Multicast FIFO

Each PHY's FIFO descriptor Table holds pointers to a structure which contains a list of channel codes that are scheduled for transmission in FIFO multicast mode (refer to Figure 39-33). The FDT resides in multi-user RAM (see Table 39-24). This FIFO Descriptor table is only relevant for FIFO WFQ working in multicast mode (see Section 39.14.3.2, "Multicast Mode Based on FIFO WFQ").

Table 39-24. FIFO Descriptor Table Entry (for Multicast mode only)

Offset	Name	Width	Description
0x00	MCST_FIFO_CC_FIRST	Word	Should be initialized by the user to point to the first CC in the CC Multicast structure for this FDT. Could be external address (if FIFO_MODES[FDT_MCST_EXT_CC]=1) or internal address. When internal mode is used it should be programmed to Multi-user RAM offset of 2 bytes only
0x04	FIFO_MODES	Hword	DMA modes and bus selection (only if using external CC structure)
0x06		Hword	Modes of operation for this Multicast FDT. See section Section 39.10.4.4, "FIFO_MODES Entry for Multicast Mode."
0x08	MCST_FIFO_CC_LAST	Word	Should be initialized by the user to point to the last CC in the CC Multicast structure for this FDT. Could be external address (if FDT_MCST_EXT_CC=1) or internal address.
0x0C	MCST_FIFO_CC_PTR	Word	Should be initialized by the user to point to the current CC in the CC Multicast structure for this FDT, which is identical to the MCST_FIFO_CC_FIRST entry initialization. Could be external address (if FDT_MCST_EXT_CC=1) or internal address.

39.10.4.4 FIFO_MODES Entry for Multicast Mode

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x04	—	—	GBL	-		—	-	BDB						-	-	
0x06	MCST_ext_cc	MCST_DISFIFO_INT_EN	FE_AUTO_SET	No_Iterations					No_Iterations_mirror					MCST_MODE	EN_FIFO	DIS_FIFO

Figure 39-40. FIFO_Modes Entry for Multicast mode

Table 39-25. FIFO_MODES Table Entry for Multicast Mode

Offset	Bit	Name	Description
0x04	0-1	-	Reserved. Initialize to zero.
	2	GBL	Global. Asserting GBL enables snooping of CC structure (if external). To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "Serial DMA Mode Register (SDMR)."
	3-6	-	Reserved. Initialize to zero.
	7	BDB	CC Structure bus (if external) 0 reside on the coherent system bus. 1 reside on the secondary bus.
	8-15	-	Reserved. Initialize to zero.
0x06	0	FDT_MCST_EXT_CC	Indicate if CC structure is external if 1, or internal if 0.
	1	FDT_MCST_DISFIFO_INTEN	Indicates if the QUICC Engine block generates an interrupt after Disabling the FIFO. In the interrupt queue entry the bits that are set are the EFCI and the MCER bits simultaneously.
	2	FE_AUTO_SET	0 Auto FE set transmitter mechanism is disabled. 1 Auto FE set transmitter mechanism is enabled.
	3-7	No_Iterations	Valid only if FE_AUTO_SET=1. Determines the number of iterations -1 on the Channel Code structure, needed by the transmitters which reside in this FIFO, to set the corresponding FE bit - if there are no valid cells to transmit.
	8-12	No_Iterations_Mirror	Should be initialized to No_Iterations. Used only by the RISC.
	13	FDT_MCST_MODE	Indicate that working in Multicast mode.
	14	FDT_EN_FIFO	Semaphore set by the core to indicate the RISC to enable this fifo again. When the Core wants to reactivate a Multicast FDT, it should assign its WFT[FTM]=0 WFT[FE]=0, and WFT[FID] =FID. The RISC resets this bit upon completion.
	15	FDT_DIS_FIFO	Semaphore set by the Core to indicate the RISC to disable this FIFO. The RISC Resets this bit to indicate the core the completion of disabling the FIFO. An interrupt could be generated if FDT_MCST_DISFIFO_INTEN is set.

39.11 UPC Policer

The MSP UPC policer implements ATM forum's TM4.1 Virtual Scheduling algorithm. Providing a dual leaky bucket policing scheme, the MSP can detect cells that violates the traffic agreement (non-conformed cells) and optionally tag (i.e changes CLP field from 0 to 1) 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 VCC Receive Connection Table (RCT) contains a pointer to a Dual Leaky bucket parameter structure (UPC parameter structure) which is residing in the external memory (Figure 39-41).

When policing has to be done for a group of VCC's (VCC's Bundle), than all the RCTs of the connections which are belonging to the same group have to point to the same UPC parameter structure. In this case the Policer controls the traffic stream generated by the unification of those VCC's.

The UPC parameter structure is depicted in Figure 39-41. The RCT[UPCPTR] field indicates the offset from UPC_BASE field in the MSP Extended Page (see Table 39-9), in which the UPC_TABLE resides in external memory. e.g. UPC_TABLE Pointer = UPC_BASE + RCT[UPCPTR]*32.

	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Offset + 0x00	**B1II[0:15]**
Offset + 0x02	**B1IF[0:11]** / **TAG1** / **LBF1**
Offset + 0x04	**B1Lim[0:15]**
Offset + 0x06	**B1Lim[16:31]**
Offset + 0x08	B1TI[0:15]
Offset + 0x0A	B1TI[16:31]
Offset + 0x0c	B1TF[0:11]] / --
Offset + 0x0e	**NCCLP0**
Offset + 0x10	**B2II[0:15]**
Offset + 0x12	**B2IF[0:11]** / **TAG1** / **LBF2**
Offset + 0x14	**B2Lim[0:15]**
Offset + 0x16	**B2Lim[16:31]**
Offset + 0x18	B2TI[0:15]
Offset + 0x1A	B2TI[16:31]
Offset + 0x1C	B2TF[0:11]] / --
Offset + 0x1E	**NCCLP1**

[1] Bold Parameters are initialized by the user. All the rest should be initialized to 0.

Figure 39-41. UPC Parameter Structure

Both leaky buckets independently performs the ATM Forum's TM4.1 **GCRA(I,L)** algorithm, where I and L express Bucket Increment and Bucket Limit respectively. In the MSP programming model, I is defined by **B1I** and **B2I** (Bucket 1 increment and Bucket 2 increment respectively). B1I (B2I) is divided to 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 PowerQUICC II Pro**CETSCR** - the **QUICC Engine Time Stamp Control Register** (See Section 20.3.9, "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.

The **LBF** (Leaky Bucket Filter) field (LBF1 for Leaky Bucket #1 and LBF2 for Leaky Bucket #2) determines the cell flow handled by the bucket (see Table 39-26 below). Both LBF1 and LBF2 have to be initialized by the user. Notice that the LBF field relates to the incoming Cells stream, i.e. if there are 2 serial Leaky buckets, the LBF field of the second Leaky bucket relates to the CLP bit of the cells that enter to the first Leaky bucket.

The **TAG** (Leaky Bucket tag) bit (TAG1 for Leaky Bucket #1 and TAG2 for Leaky Bucket #2) determines bucket's operation for non conformed cells, according to Table 39-27..

The MSP Policer supports Packet oriented discard mode (PPD). This mode is enabled by setting **RCT[PPDE]** bit. When this mode is enabled and a single cell is dropped by the policer, then all other cells associated with the current packet are dropped. The last cell of the packet (PTI[0]=1) is policed normally.

39.11.1 UPC Programming

Before enabling the UPC for any channel, the CETSCR must be initialized. The CETSCR defines the time unit which is common to all leaky buckets and Time Stamps in the PowerQUICC II Pro. Following that, each individual leaky bucket is programmed independently.

Each one of the dual leaky buckets has four user defined parameters:

- LBF - Leaky Bucket filter. (See Table 39-26)
- TAG/Discard bit (See Table 39-27)
- Bucket Increment (BI)
- Bucket Limit (BLim)

The LBF field and the TAG bit are programmed according to the definitions in Table 39-26 and Table 39-27 respectively.

Table 39-26. LBF—The Leaky Bucket Filter

Cell type monitored	LBF coding
Disable policing	000
CLP0 cell policing. CLP1 cells are bypassing this bucket	001
CLP1 Cell Policing. CLP0 cells are bypassing this bucket	010
CLP0+1 Policing. All cells are tested by this bucket	011
F-GCRA Policing. Only First cell of each packet is tested for conformance. All the rest of the cells gets the same conformance result as first cell.	100
Reserved	101
Reserved	110

Table 39-27. TAG Bit Interpretation

Cell type monitored	TAG
Drop non-conformed cells	0
Tag non-conformed Cells	1

The formula for the decimal representation of the Bucket Increment is:

A) BI = CellSize/(BitRate x TimeUnit)

Where **BI** is the decimal expression for the bucket's increment, **CellSize** is the size of a cell on the physical media considered (in units of bits), and **TimeUnit** is the time interval of one count of CETSR. Since CETSR 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 has to be converted to hexadecimal Fixed Point representation.

Example

We have to program the leaky bucket for a connection with rate of 16Kbps, where the fastest connection in the system is 155Mbps. Let's find the value of BI, assuming cell size of 53 bytes.

B) TimeUnit <= 53x8/155Mbps = 2.735us

So CETSR is programmed to, say, 2.7us.

Substituting the value in (A) we get:

BI = (53x8)/(16kbps x 2.7us) = 9814.8148

Converting it to Hexadecimal, we get: BI = 0x2656.D09. This implies the following programming:

B1II[0:11] = 265, B1II[12:15] = 6, B1IF[0:11] = D09

The smallest data rate which is given by the substitution of BI=64K in (A), is 2400bps. The resolution is at least 0.02%.

The formula for the Bucket Limit is:

C) BLim = CDVT/TimeUnit

Where **BL** is the decimal expression for the bucket's Limit, and **TimeUnit** is the time interval of one count of CETSR. For details about TimeUnit definition please refer to the explanation about BI programming above.

Following the calculation in (C), the result is rounded to the closest integer and converted to hexadecimal representation. The smallest value of L is 1 TimeUnit (2.7us in the above example). The highest value is 190min.

39.11.2 Example For Using PCR, CDVT

Highest Frequency: 16Mbps

PCR=1536Kbps

CDVT=0.5sec

TimeUnit <= 53x8/16Mbps = 26.5us.

This value of TimeUnit is larger than the highest possible 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

39.11.3 Relationship Between Two Instances of GCRA(I,L)

Up to two instances of Leaky Bucket with a different pairs of Bucket Increment (BI) and Bucket Limit (BL) might be applied to the data belonging to the same VCC, VPC, or any arbitrary bundle of VCCs.

The Leaky Bucket pair implies a certain coupling between the buckets. The MSP supports Leaky Bucket instances coupling according to the ITU-T recommendation I.371 and ATM Forum TM4.1 Specifications.

Each of the leaky buckets might be programmed to test conformance of CLP0, CLP1 or CLP0+1 cell streams. A cell which is tested by both buckets, is conformed only if it found conformed by both of the leaky buckets. The bucket state is updated only if a cell is conformed. Only a leaky bucket that found a cell conformance updates its state.

The exact behavior of Two Leaky Bucket instances is depicted in Figure 39-42 and Figure 39-43 below.

In the flows, a cell is processed by a certain instance of a leaky bucket if the LBF field selects a certain cell for this bucket. If a cell is conformed, this bucket will update its state if the cell will be accepted.

For F-GCRA (LBF=100), the conformance test is done only for the first cell of a packet. All the rest of the cells in the packet are processed by the Leaky Bucket, but assume conformance result similar to that of the first cell.

When the PPDE (Partial Packet Discard Enable) in the RCT is set, then the UPC policer is working at PPD mode. In this mode, if a cell is dropped, the rest of the cells in the packet except for the last one are dropped as well, and assume the same policing result as the first dropped cell.

Some Leaky Bucket instances examples are shown in Table 39-28. The examples show Leaky Bucket programming for all ATM Forum TM Specification 4.1 service Categories and ITU-T I.371 ATM Transfer Capabilities.

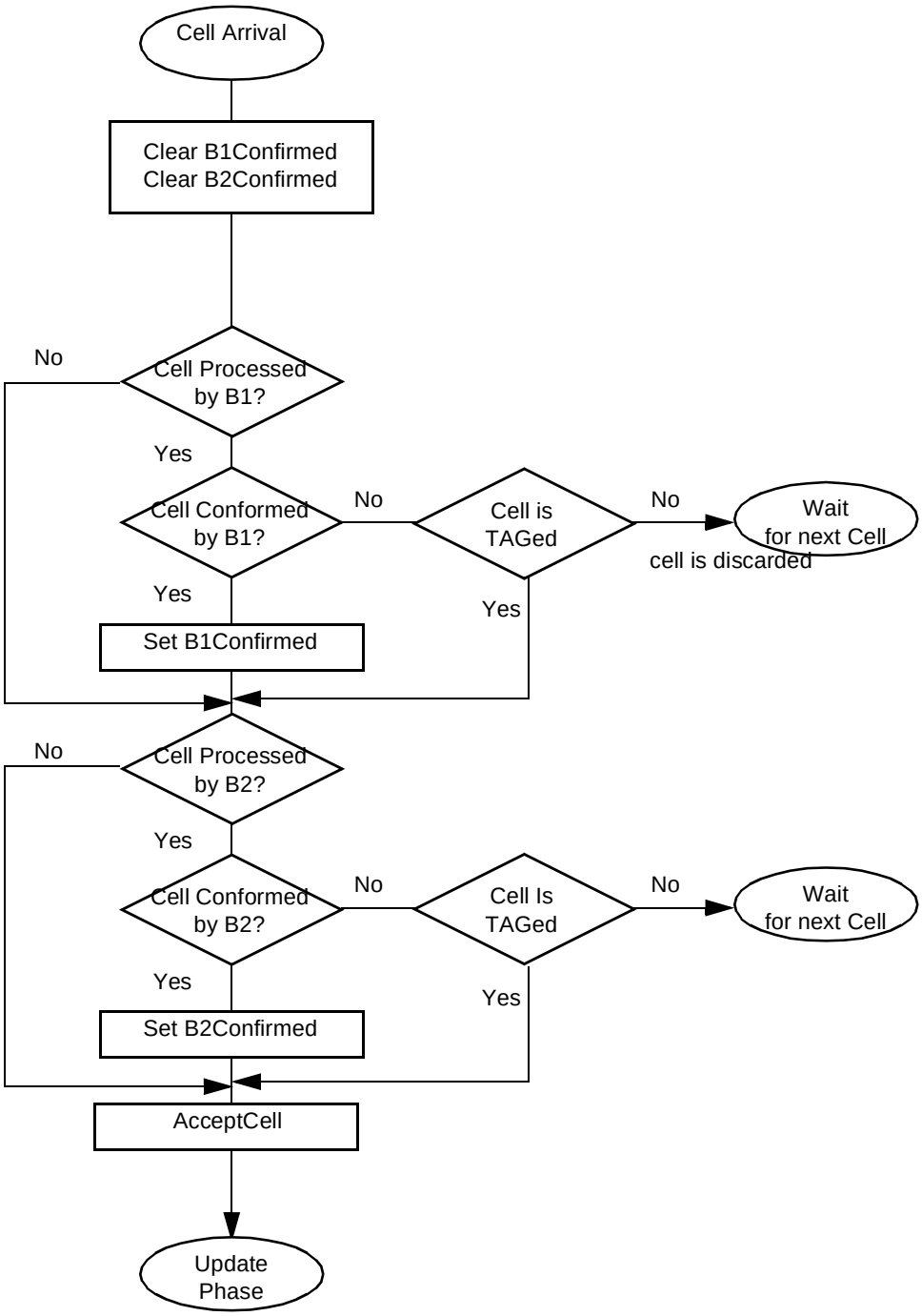


Figure 39-42. Coupling Between Two Leaky Buckets

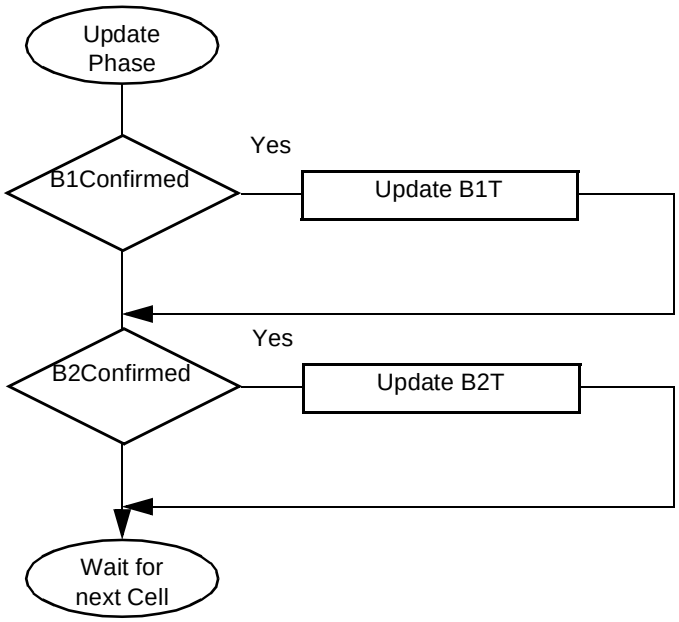


Figure 39-43. Updating B1T and B2T

39.11.4 GFR Conformance

The MSP supports GFR conformance testing according to the ATM Forum TM4.1 Specifications for GFR.1 and GFR.2 service categories. A frame is conforming if all its cells are conforming. GFR conformance implies the following conditions:

(1) The cell conforms to GCRA(1/PCR, CDVT) for the CLP0+1 cell stream

(2) The CLP value of any cell in the frame is identical to the CLP in the first cell

(3) The frame is shorter or equal to the MFS (Maximum Frame Size)

Condition (1) is tested by programming a leaky bucket to test flow CLP0+1, and drop any non-conformed cell. If a cell is dropped, a PPD policy is taken for the rest of the frame.

For Condition (2) testing, two user defined policies are available in GFR mode. The first is to discard any cell which has a CLP value different than the value of the CLP in the first cell. In this case, a PPD policy is taken for the rest of the frame. The second policy is similar to the first, except that for frames in which the first cell has a CLP value of 1, all the rest of the cells are tagged.

Condition (3) is enabled while GFR mode is selected in the RCT. The accumulated frame length is compared to a threshold, GFRT, which is located in the RCT. If the accumulated length is bigger then the threshold, a PPD policy is taken for the rest of the frame.

39.11.5 GFR Frame-Eligibility

A frame is said to be “eligible” if and only if it is conformed, un marked, and passes the F-GCRA test (ATM Forum TM specifications Version 4.1 Annex B.5). The agreed QoS level is guaranteed only for eligible frames.

The MSP Policer supports the F-GCRA. Selecting the F-GCRA mode is done by programming the filter LBF to be 100 and TAG=1 (See Table 39-26 and Table 39-27).

In this mode, the F-GCRA(1/MCR, BT+CDVT) Leaky Bucket (Typically the second Leaky Bucket in double leaky bucket instances) will TAG a complete frame if it finds the first cell of the frame non eligible, and change frame’s eligibility status.

Another case where a status of a frame can be changed from non-conformed to non-eligible is when the CLP test marks the cells in a frame which has its first cell marked.

A non-eligible frame can be either dropped or getting a lower level service in the Queue Management (See Section 39.8, “Queue Management). If the network requires to drop non-eligible frame, it can be done by programming the TAG bit in the F-GCRA Leaky Bucket to 0 (Non TAG mode).

Table 39-28. Example to Double Leaky Bucket Configurations

ATM Forum TM4.1 Service Categories	ITU-T I.371 ATM Transfer Capability	LB1		LB2	
		LBF	TAG	LBF	TAG
CBR.1	DBR Configuration 1	011	0	000	0
VBR.1	SBR Configuration 1	011	0	011	0
VBR.2	SBR Configuration 2	011	0	001	0
VBR.3	SBR Configuration 3	011	0	001	1
GFR.1	—	011	0	100	0
GFR.2	—	011	0	100	1
UBR.1	—	011	0	0	0
UBR.2	—	011	1	0	0

39.12 Billing

In order to support a billing mechanism the MSP performs cell counting for Ingress/Egress connections.

For Ingress connections the billing is performed for all incoming switched cells (as described in the general Ingress Data Flow, see Section 39.1.1, “Ingress Data Flow”), therefore the cell counting is done in the SRCT (See Section 39.9.2, “SRCT - Switched Cells Receive Parameter Table”). For Egress connections billing is performed for all transmitted cells, therefore the cell counting is done in the STCT (See Section 39.9.4, “STCT - Switched Cells Non Hierarchical Scheduling).

For both Ingress/Egress connections two Cell counters are supported, 32bit CLP0 count and 16 bit CLP1 count. Whenever one of these counters overflows, a maskable interrupt is generated to the host.

39.13 Statistics

In Addition to the Ingress/Egress Cell counters used for billing purposes, the MSP manages a set of counters for statistics, on a per connection basis.

Note that the CLP0 and CLP1 Cell count (as explained in the Billing section) is performed always at both receiving and transmitting sides.

If the UPC policer is enabled (on a per connection basis) additional two counters are introduced: non conforming CLP0 counter (NCCLP0) and non conforming CLP1 counter (NCCLP1). Both are 16-bit counters and reside in the UPC parameter table (see Figure 39-41).

An additional counter QDCC (resides in the SRCT) is used for monitoring the number of cell which were dropped by the Policing/ Queue management mechanism. Whenever a cell is dropped by the Queue management mechanism, this counter is updated. Whenever one of these counters overflows, a maskable interrupt is generated to the host.

39.14 Multicast

The QUICC Engine EMSP is capable of performing **MULTICAST** operation.

When the cells which are being received over a certain connection has to be duplicated and transmitted over an arbitrary set of other connections and PHYs transmission path, a **MULTICAST** operation is executed.

39.14.1 Definitions

The cell which has to be duplicated is received by the QUICC Engine EMSP through the S**ource Connection**, and sent out through a group of **Destination Connections**.

The connection code (CC) of the receiving cells is called **Source Connection**, the group of TCCs of the connections which the cell has to be duplicated over is called the **Destination Group**. Each **Destination Group** contains two or more **Destination Connections** up to 32 connections are available.

If, for instance, the cells received over CC #1000 has to be transmitted to connections with Channel Codes (CCs) 2000, 3000, 4000 then CC #1000 is the Source CC and CCs 2000, 3000 and 4000 are the members of the destination group.

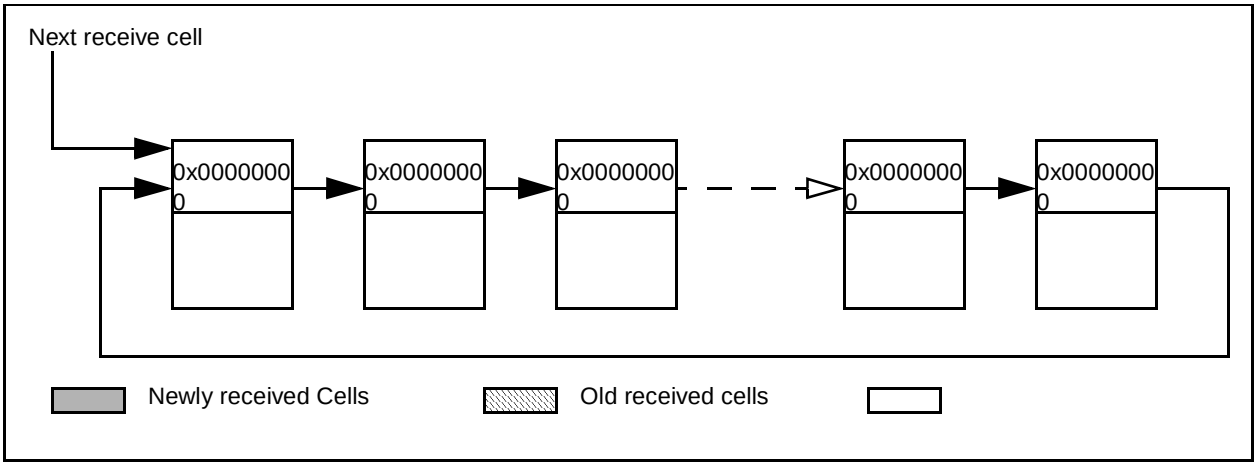
The **Source connection**'s linked list of received cells is called **Source Queue.**

The above terms and the multicast implementation are depicted in Figure 39-44 below.

39.14.2 Multicast Source Queue

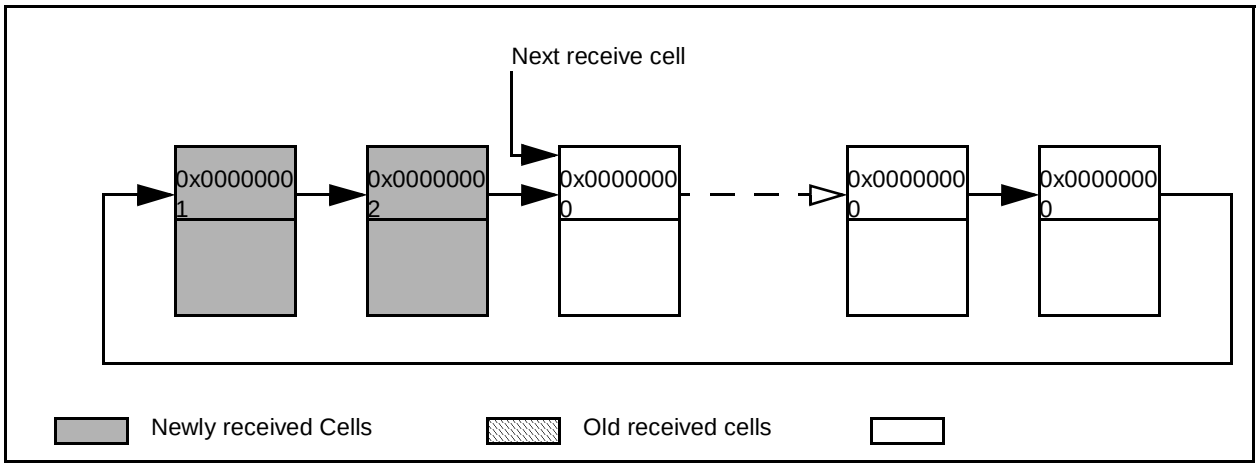
All multicast cells (cells which have RCT[RCTM]=11) are stored in a special multicast queue. There is one queue for each multicast group; the queue is common to all connections and all PHYs. The length of the queue is determined by the application SW, and should be larger than the maximum delay (in cell time) between receiving and transmitting a cell to an output port. The Multicast Source Queue is a cyclic queue, therefore, new incoming cells will override older cells after the first wrap of the Queue. Depicted below is an example of an initialized Multicast Source Queue, which size is 1000(hex) cells. All of the cells are

empty and have a serial number of 0x00000000. The Queue is allocated and initialized by the application SW.



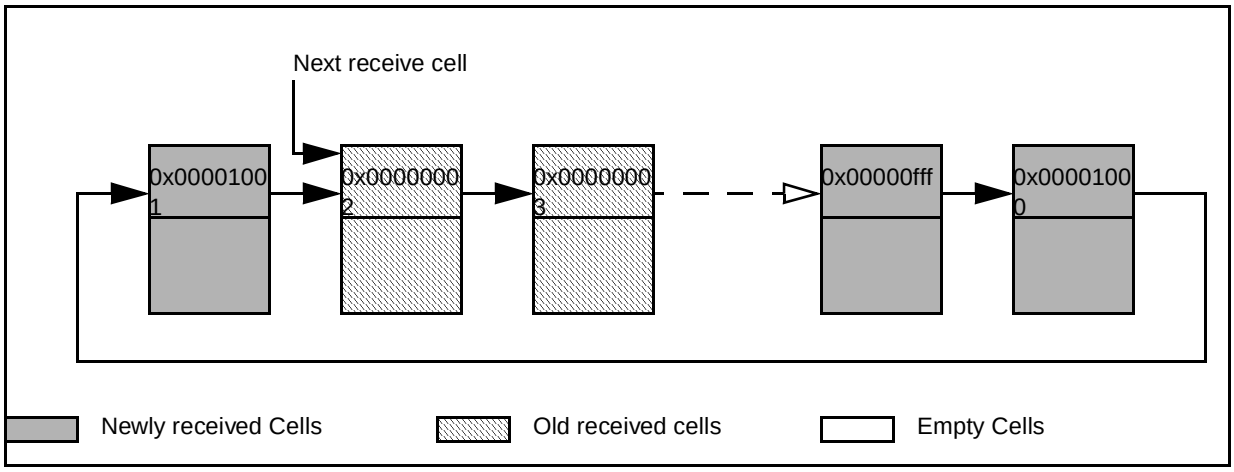
The ‘Next receive cell pointer’ points to a ‘free’ cell to which the receiver will accept a new incoming cell. This pointer should be initialized by the host SW to point to any of the free cells in the cyclic Source Queue.

After receiving two cells, the Queue should look like this:



Note that each received cell is associated with a serial number. The purpose of the serial number will be explained later.

After receiving 1001 cells, the Queue should look like this:



It is easily seen that new incoming cells override older cells, which should have been transmitted already by all of the Destination members.

39.14.3 Multicast Operation Modes

39.14.3.1 Multicast Mode Based on APC Shaping

When working with Multicast using APC Shaping (MTCT[FIFO_MODE]=’0’), the Multicast transmitter members are chosen using the APC of each PHY. This enables that transmitters from different PHYs could use the same Multicast source list. it’s the user’s responsibility to issue a host command to activate each member of the Destination Group in the APC.

This mode enables that on the same PHY there could be unicast APC connections and at the same time Multicast APC connections.

39.14.3.2 Multicast Mode Based on FIFO WFQ

When working with Multicast using FIFO WFQ (MTCT[FIFO_MODE]=’1’), the multicast transmitter members are chosen using the FIFO WFQ of each PHY. When a ‘FIFO’ is chosen by the FIFO WFQ algorithm, and it is a multicast FDT, then the CC are chosen using the CC structure pointed by this FDT by round robin algorithm (see Figure 39-33) Then, the MTCT is read instead of the FTCT, to identify which linked list this channel code belongs to. This enables that transmitters from different PHYs could use the same Multicast source list.

This mode enables that on the same PHY there could be unicast FIFO WFQ connections and at the same time Multicast FIFO WFQ connections.

39.14.3.3 Multicast Operation

After cell processing by the receiver, the cell is placed in the appropriate multicast Queue. Each one of the Destination Connections is pointing to the next cell to be transmitted from the multicast Queue. When a cell is transmitted by a member of the destination group, the member’s pointer is changed to point to the

next cell in the multicast Queue. Each destination group member has its own pointer to the multicast queue, so there is a complete isolation between the destination connections. If one destination member fails to perform transmit operation from some reason, all other destination group members continue to transmit normally.

When receiving a new cell to the Source Queue, the receiver will write two additional fields to the COV area of the cell. One is an incremental serial number, copied from the MRCT and the other is the Multicast members mask (also copied from the MRCT).

Each destination group member keeps a serial number counter. When attempting to transmit, this serial number should match the one in the COV of the cell to be transmitted. The transmitter compares these two serial numbers. If they match, then the transmit operation will continue normally. If a mismatch occurs there are two possible reasons for this to happen: If the transmitter is faster then the receiver, the SN- serial number of the cell is smaller then the expected SN of the transmitter, the transmitter could try to transmit a cell that already has been transmitted (See above figures describing the Cyclic Source Queue). The use of the serial numbers prevents this from happening by causing the transmitter to stop and wait until new cells are received. Alternatively, if for some reason a destination member could not perform transmit operation for a long time and its pointer to the Multicast Queue is no longer valid (i.e the cell to which the MTCT points was already overwritten by a new cell thus the cell SN is bigger then this in the MTCT), a mismatch will occur between the transmit serial number in the MTCT and the receive serial number stored in the COV area. In this case the behavior is programmable. If the MTCT[AVCOF]=’1’ then the destination channel could be automatically removed from being scheduled by the APC, or in case of FIFO WFQ mode the transmitter will wait endlessly, and the user should remove it from the corresponding CC structure. If MTCT[AVCOF]=‘0’, then the transmission continues and the new serial number is adopted by the transmitter. In either case, the RISC will generate a maskable MCET error interrupt to the host indicating to the application which of the destination channels was stuck. Adding the channel back should be performed by the host as described in Section 39.14.5, “On-Line Modification of the Multicast Destination Group.”

In order to alert the host SW that one or more of the destination group members stopped performing transmit operation, an additional mechanism is being used. This mechanism is necessary in cases where one or more of the destination members is stuck for a long period of time. Each of the destination group members has an I.D number ranging from 0-31. This unique number is determined in the MTCT[MCIDN]. In normal operation each destination member, upon transmission of a cell, clears the bit associated with it’s ID number in the Multicast members field in the COV area of the cell (see Figure 39-47). For example a destination channel with MCIDN = 8 will clear bit 8 in the cell COV[Multicast members] word. It is expected that upon transmission of a cell by all of the destination members this word is zero.

The source connection is aware of the number of destination group members. This is achieved by the Multicast Members Mask word in the MRCT. This is a mask where each bit represents a destination member ID. In normal operation, upon receiving a cell the receiver check that the [Multicast members] word in the COV area of the current cell location is zero. This implies that the cell was transmitted by all of the destination group members. The receiver detects a situation in which an old cell that is going to be overwritten and generates a maskable MCER interrupt to the host SW. In either case the source channel updates the Multicast Members field in the COV area to the Multicast Members Mask in it’s MRCT, and writes the new cell to this location. The information regarding which of the destination channels is “stuck” is described in Section 39.14.6, “Multicast Receiver Event.”

Also note that because the Source Queue uses a different mechanism than a Unicast VC or VP Queue, NO other Queue management is applicable (The Unicast Queue Thresholds do not apply for a Multicast connection).

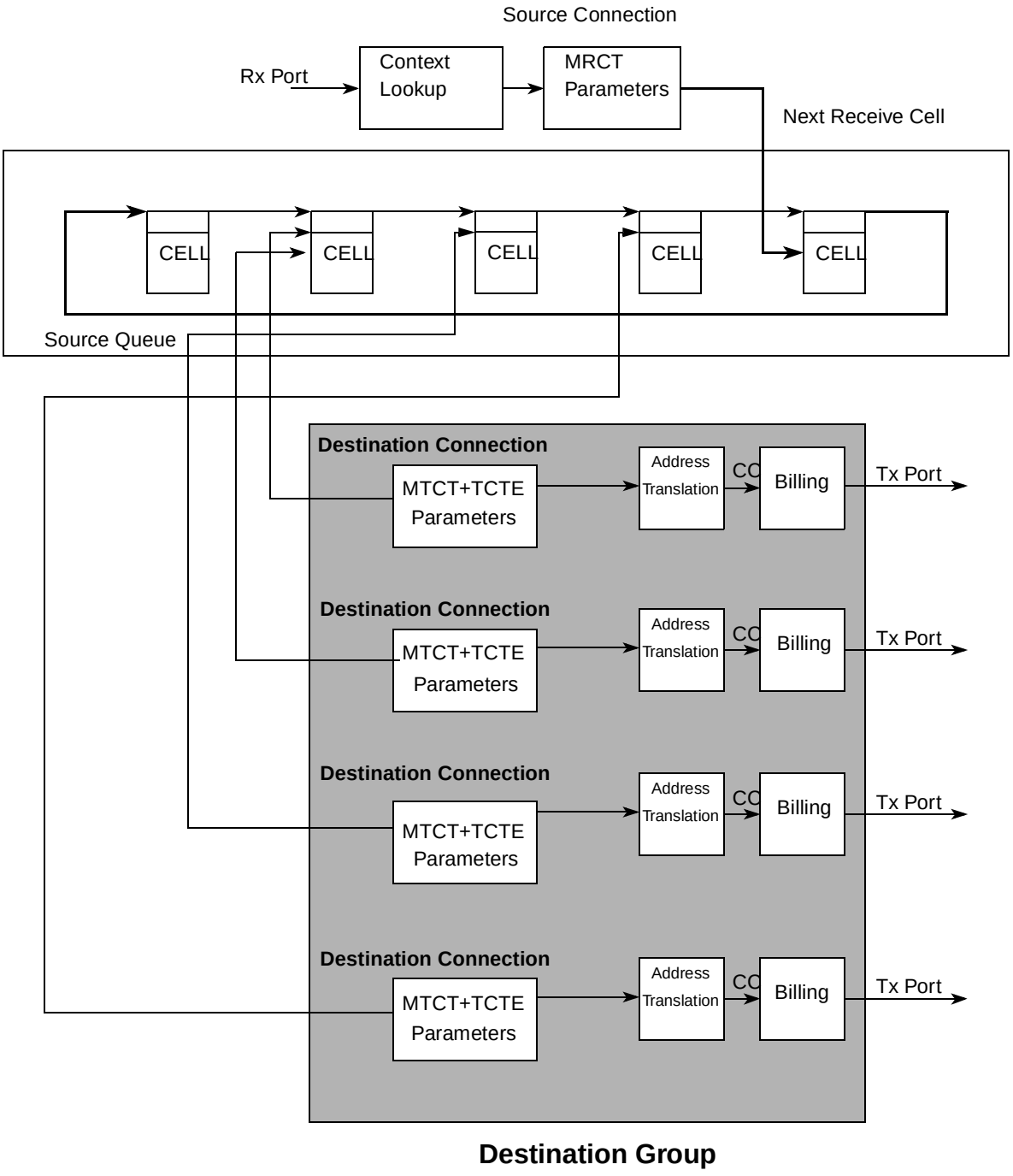


Figure 39-44. Multicast: Address Translation and Billing is Done Independently for Each Member in the Destination

39.14.3.4 Auto FE mechanism for FIFO Multicast Mode

The Auto FE set/clear mechanism for multicast FIFO operation, is enabled by setting MRCT[FE_AUTO_CLR], FDT[FIFO_MODES]: FE_AUTO_SET and NO_ITERATION fields. This mode is designed to emulate FIFO Empty mechanism, for applications that use Strict/Fix priority modes, for example. This mode is the preferred mode of working, when using FIFO mode scheduling for Multicast connections.

When auto FE mechanism is on, on each receive of cell using MRCT, the RISC reads the Pointer to the WFT Pointer Table from FEPTR_BASE + MRCT[MCGID]*2 offset in multi-user RAM. This WFT Pointer Table consists of all the WFTs entries needed to be accessed by the RISC, in order to clear their corresponding WFT[FE] bits. This WFT Pointer Table also resides in multi-user RAM. The COUNT in the beginning of this table indicates the number of WFT entries minus 1. See Figure 39-45.

On the transmitter side, each time that a MTCT is selected from a certain FIFO, and there is no valid cell to transmit from its Multicast list, then the FDTs FIFO_MODES[NO_ITERATION] field, is used to determine when to set the corresponding FE bit, see Section 39.10.4.4, “FIFO_MODES Entry for Multicast Mode”. Only after the Channel Code structure was scanned NO_ITERATION + 1 times, and still there is no valid cell to transmit, then the corresponding FE bit is set and the FIFO is disabled.

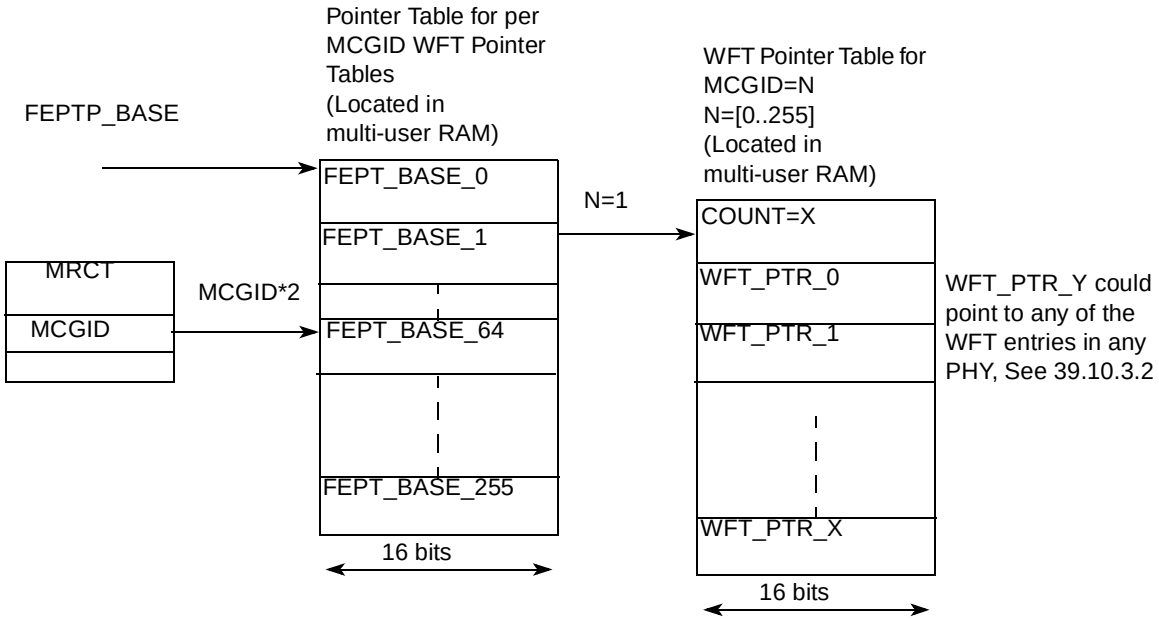


Figure 39-45. WFT Pointer Table

39.14.4 MSP Multicast Main Features

- 32 channels of Destination Connections per source connection.

Note - If the user wants more than 32 Destination group per source connection, it is possible to use the same MCIDN value for all Channels which reside in the same PHY - this way the user could still have the

knowledge of which PHY failed on a multicast receiver event, see Section 39.14.6, "Multicast Receiver Event."

- 256 available source channels per UCC.
- Per Destination CLP1 and CLP0 statistics.
- Arriving cells are stored in a queue per destination group.
- User Defined Address Translation for each Destination Connection Independently
- Cells are not duplicated in Memory.
- No Extra performance cost except for regular transmission performance cost per additional Destination Group member.
- Supports On-Line (Run-Time) Destination-Group modification.

39.14.5 On-Line Modification of the Multicast Destination Group

On line modification enables the application to exclude a member or include a new member in the Destination Group without interfering with the multicast operation. It is important to notice the difference between On-Line modification and normal power up setup:

When setting up the connections on power up, the following steps are performed:

- In order to avoid the receiver from start receiving cells before the Transmitters are configured, the user should disable the receiver by setting the MS bit in the corresponding CAM or Address Compression output entry, or set the MRCT[MS], when working with CC in UDH mode.
- MTCT[MQPTR]=MRCT[Next Receive Cell Pointer], MTCT[SN]=MRCT[SN]=0, MTCT[ADDCH]=0.
- For FIFO non-auto FE mode, clear the FE bit in the WFT entry. For FIFO with auto FE mode - set the FE bit.
- For APC mode issue an ATM transmit command.
- Enable the MRCT connection, by clearing the MS bit.

For dynamic modification see the following sections.

39.14.5.1 On-Line Modification of Multicast APC Mode

Adding a destination member to the destination group is done in the following way:

1. Allocate MTCT entry for the new channel with MTCT[TCTM] =11.

2. Set the MTCT pointer to point to the Cell in the linked list which is not currently pointed by the MRCT. i.e. the MTCT pointer should point to a cell which lags the source pointer by at least one cell. This, in order to prevent synchronization problems between Rx and Tx. Leave the SN entry in the MTCT uninitialized.

3. Set the ADDCH bit in the MTCT. MTCT[SN] field value would be taken from the COV of the cell and would be adopted automatically by the RISC- so this field could be cleared.

4. Update the Multicast Member Mask field (word) in the MRCT of the source queue.

5. Issue an ATM Transmit host command to enable the scheduling of the channel in the APC.

Upon completion of the insertion of this new destination member the ADDCH bit in MTCT is cleared.

Removing a destination member from the destination group is done in the following way:

1. Update the Multicast Member Mask field (word) in the MRCT of the source queue.

2. Set the STPT bit in the MTCT of the destination member to be removed.

39.14.5.2 On-Line Modification of FIFO WFQ Multicast Mode without Auto FE Mechanism

Adding a destination member to the destination group is done in the following way:

1. Setting the FDT[MCST_FIFO_DIS] bit of the Multicast FIFO.
2. Wait till FDT[MCST_FIFO_DIS] is cleared or till an interrupt is issued, with bits EFCI and MCER set to 1, depending on FDT[FDT_MCST_DISFIFO_INTEN] (see Figure 39-25
3. Add the required channel code to the Channel code structure of the Multicast FIFO.
4. Allocate MTCT entry for these new channels with MTCT[TCTM] =11, MTCT[FIFO_MODE]=1 and set the ADDCH bit. MTCT[SN] field value would be taken from the COV of the cell and would be adopted automatically by the RISC- so this field could be cleared.
5. Set the MTCT pointer to point to the Cell in the linked list which is not currently pointed by the MRCT. i.e. the MTCT pointer should point to a cell which lags the source pointer by at least one cell. This, in order to prevent synchronization problems between Rx and Tx. Leave the SN entry in the MTCT uninitialized.
6. Update the Multicast Member Mask field (word) in the MRCT of the source queue.
7. Activate the Multicast FIFO by setting the WFT of the corresponding FIFO to the following value: WFT[FE]=0, WFT[FT]=0 and WFT[FID]=FID, and after that setting FDT[MCST_FIFO_EN] bit to 1. The reason for setting ‘FIFO_EN’ bit, is to enable the UP to update the current minimum WFT to the added FIFO. The RISC will reset this bit upon completion.

Upon completion of the insertion of the new destination member the ADDCH bit in the MTCT is cleared. It’s possible to add some channel codes which belong to the same FIFO (the same CC structure) simultaneously.

Removing a destination member from the destination group is done in the following way:

1. Setting the FDT[MCST_FIFO_DIS] bit of the Multicast FIFO.
2. Wait till FDT[MCST_FIFO_DIS] is cleared or till an interrupt is issued, with bits EFCI and MCER set to 1, depending on FDT[FDT_MCST_DISFIFO_INTEN] (see Table 39-25, “FIFO_MODES Table Entry for Multicast mode.”
3. Remove the required channel codes from the Channel code structure of the Multicast FIFO.
4. Update the Multicast Member Mask field (word) in the MRCT of the source queue.

If the Channel Code structure isn’t empty at this point, then step no. 5 should be performed, in order to reactivate this FIFO WFT mechanism. On the Contrary, if the Channel Code structure is empty, then the following step shouldn’t be performed.

5. Activate the Multicast FIFO by setting the WFT of the corresponding FIFO to the following value: WFT[FE]=0, WFT[FT]=0 and WFT[FID]=FID, and after that setting FDT[MCST_FIFO_EN] bit to 1. The RISC will reset this bit upon completion.

It is possible to remove some channel codes which belong to the same FIFO (the same CC structure) simultaneously.

39.14.5.3 On-Line Modification of FIFO WFQ Multicast Mode with Auto FE Mechanism

The mechanism of On Line modification described in the previous section, must be changed for Auto FE FIFO Multicast mode. This is because the manual enable/disable operations, that changes the value of the FIFO FE bit, can interfere with the automatic set/clear of the same FE bit. Therefore the user shouldn't use the FIFO_DIS operation anymore.

The sequence of adding a CC to the destination group is -

1. Allocate MTCT entry for these new channels with MTCT[TCTM] =11, MTCT[FIFO_MODE]=1 and set the ADDCH bit. MTCT[SN] field value would be taken from the COV of the cell and would be adopted automatically by the RISC- so this field could be cleared.
2. Set the MTCT pointer to point to the Cell in the linked list which is not currently pointed by the MRCT. i.e. the MTCT pointer should point to a cell which lags the source pointer by at least one cell. This, in order to prevent synchronization problems between Rx and Tx. Leave the SN entry in the MTCT uninitialized.
3. Update the Multicast Member Mask field (word) in the MRCT of the source queue.
4. Modify the CC structure, so that the new CCs would be added to the end of the list.
5. Modify the MCST_FIFO_CC_LAST in the FDT, see Section 39.10.4.3, "FDT for Multicast FIFO."
6. Update the No_Iterations field in the FDT, if necessary.
7. If it is a new CC leaf of a specific MRCT, then the user should update the MRCT's WFT Pointer Table, so now the entry for this FIFO's WFT will be added- see Section 39.14.3.4, "Auto FE mechanism for FIFO Multicast Mode"
8. Enable the FIFO if it is the first CC in the CC list, by setting WFT[FT]=0,WFT[FID]=FID (no need to clear WFT[FE] bit) and setting FDT[FIFO_EN]. The reason for setting 'FIFO_EN' bit, is to enable the UP to update the current minimum WFT to the added FIFO. The RISC will reset this bit upon completion.

The sequence of deleting a CC from the destination group is -

1. Update the Multicast Member Mask field (word) in the MRCT of the source queue.
2. Rearrange the CC structure such that removing the CC, which is intended to be deleted, wouldn't create spaces ('holes') in the CC list.
3. If the Channel to be removed is the last Tx destination member of a specific MRCT in this FIFO, then update the MRCT's WFT Pointer Table, so now the entry for this FIFO's WFT will be deleted - see Section 39.14.3.4, "Auto FE mechanism for FIFO Multicast Mode"

If it is the last CC deleted from the CC list then perform stages 4-5 -

4. Update the MCST_FIFO_CC_LAST field in the FDT to be equal to the MCST_FIFO_CC_FIRST value.
5. Set the Last CC value to zero and the No_Iterations field in the FDT to zero. This will cause the FIFO to be disabled automatically. Wait till the FE bit in the corresponding WFT entry will be actually set by the RISC. NO need to set FDT[FIFO_DIS] bit since it isn't supported for AUTO_FE mode.

If it is NOT the last CC deleted from the CC list then perform the following stages -

6. Update the MCST_FIFO_CC_LAST in the FDT, which will prevent the deleted CC to be scanned.
7. Update the No_Iterations field in the FDT, if necessary.

Notice - There could be conditions where the receiver stopped receiving anymore cells and will not receive anymore cells to a specific FIFO, and therefore it doesn't clear the FIFOs WFT[FE] anymore, but the FIFO transmitters didn't transmitted all the cells yet, since they set WFT[FE] bit already. In this case the user should set WFT[FE]=0 (FDT[FIFO_EN] is already set in that condition by the RISC), so the transmitters of that FIFO will transmit the last cell it the multicast source list.

39.14.6 Multicast Receiver Event

In case when a source channel detects that not all of the Multicast members field in the COV area of the cell were cleared by the transmitter member group, and the MRCT[MCRIE] is set, an event indication is set in the interrupt queue entry. The information given in such an interrupt entry is the receiver channel code causing the event. Further info is required to determine which of the destination channels is overwritten by the receiver and the information is obtained in this way: In the MSP parameter table the entry MCE_BASE points to the Multicast Event Table structure which consists of up to 256 one word entries. Each of the source channels of the multicast is assigned with a group number MRCT[MCGID]. So each of the entries in this table corresponds to a source group by the MCE_BASE address + MCGID*4. Each entry in the table is a replication of the multicast Member field in the COV area which was overwritten by the source channel. The user can tell which of the destination channels was overwritten since the bit representing it's ID is set.

Table 39-29. Multicast Event Table

Address	Name	Width	Description
Offset+ 0x00	Multicast Member group 0	word	Multicast Member image for source group 0
Offset+ 0x04	Multicast Member group 1	word	Multicast Member image for source group 1
Offset+ 0x04*i	Multicast Member group i	word	Multicast Member image for source group i
Offset+ 0x3FC	Multicast Member group 255	word	Multicast Member image for source group 31

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

Figure 39-46. Multicast Member group i

39.14.7 Multicast Cells Format

At initialization, each MRCT[Next_Receive_Cell_Ptr] and MTCT[MQPTR] should point to a free cell of the following format. All fields are given for informative purposes only.

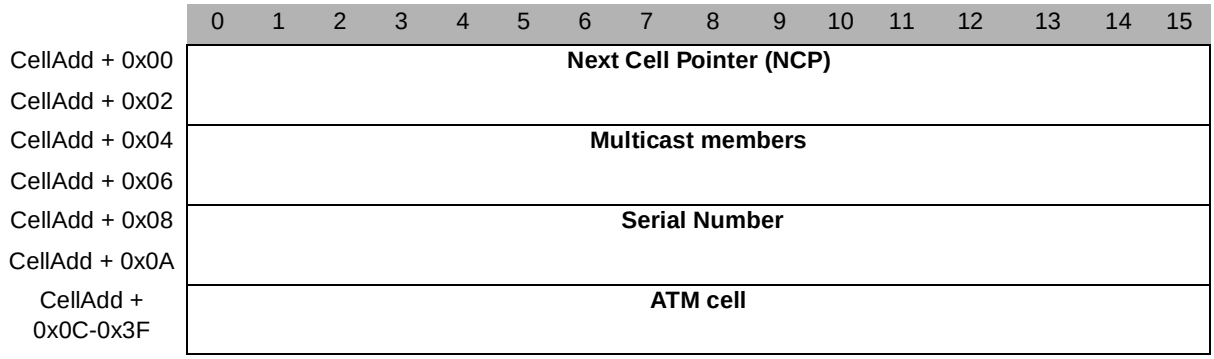


Figure 39-47. Multicast ATM Cell Format

The Serial Number (SN) is updated by the receiver for each incoming cell.

The Multicast Members field is a mask for 32 available destination channels used by the receiver to detect whether one or more of the destination connections stopped performing transmit operation.

The Multicast members and Serial Number fields should be cleared and are for RISC use only. The NCP field should be initialized to point to the next cell in the source link list.

It is preferred that the cells in the Multicast linked list would be ordered one after the other in the external memory. This in order to simplify the action of finding the cell in the linked list which lags the source pointer by at least one cell, as described in Section 39.14.5, “On-Line Modification of the Multicast Destination Group” for adding a destination member to the multicast group. In this case the only operation needed to be done for finding the MQPTR of the added MTCT, is to substitute 0x40 from MRCT[next_receive_cell_ptr], or in case that MRCT[next_receive_cell_ptr] points to the beginning of the linked list, MTCT[MQPTR] should be the address of the last cell in the linked list.

39.14.8 MRCT- Multicast RCT

RCTM=11

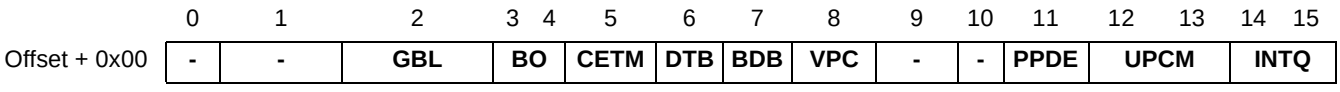


Figure 39-48. Multicast Receive Connection Table (MRCT)

Offset + 0x02	RCTM		FE_AUTO_CLR	—	-	MCGID							AAL
Offset + 0x04 Offset + 0x06	Receive Cell Serial Number												
Offset + 0x08 Offset + 0x0A	Cell Time Stamp												
Offset + 0x0C	QDCCt												
Offset + 0x0E	CLP1 Count												
Offset + 0x10 Offset + 0x12	CLP0 Count												
Offset + 0x14 Offset + 0x16	Next Receive Cell Pointer												
Offset + 0x18	UPCPTR												
Offset + 0x1A Offset + 0x1C	Multicast Members Mask												
Offset + 0x1E	MS	NCCLPIE	PMT				SEGT	ENDT	-	-	MCRIE	QDCCIE	CLPIE	PME

Figure 39-48. Multicast Receive Connection Table (MRCT)

Table 39-30 describes Multicast RCT fields.

Table 39-30. Multicast RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0		Reserved. Set to zero.
	1		Reserved. Set to zero.
	2	**GBL**	Global. Asserting GBL signal on memory bus, 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 19.1.8.2, "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.
	6	**DTB**	Data buffers bus (source linked list cells) 0 Cell buffers reside on the coherent system bus. 1 Cell buffers reside on the secondary bus.
	7	**BDB**	Interrupt queues and UPCT 0 Interrupt queues and UPCT reside on the coherent system bus. 1 Interrupt queues and UPCT reside on the secondary bus.
	8	**VPC**	Virtual Path Connection (used for OAM termination) 0 This channel is not a VP connection 1 This channel is a VP connection
	9-11	**-**	Reserved. Initialized to zero.
	11	**PPDE**	Partial Packet Discard (PPD) Enable. When a single cell is dropped by the policer, then all other cells associated with the current packet are dropped. The last cell in a packet will always be accepted. Only if the first cell is discarded then the Last cell will be discarded as well. Setting this bit implies frame based connection (AAL5). 0 PPD is disabled. 1 PPD is enabled. It is preferred from performance issues that PPDE will be disabled, even if working in frame mode, when the policer is disabled - since in this case no cells should be dropped.
	12-13	**UPCM**	UPC (Policing) Mode (See Section 39.11, "UPC Policer" for details) 00 UPC is disabled. 01 UPC is enabled. 10 Not valid for Multi-cast mode. 11 Not valid for Multi-cast mode.
	14–15	**INTQ**	Points to one of four interrupt queues available.

Table 39-30. Multicast RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0-1	**RCTM**	RCT Mode 00 This RCT is used for a switched channel shaped by the APC. 32 bit QPTRL is placed in address 0x4 of RCT. 01 This RCT is used for a switched channel shaped by external FIFOs. 32 bit EFDP FIFO Descriptor Pointer to external memory is placed in address 0x4 of RCT. 10 This RCT is used for a switched channel shaped by internal FIFOs. 16 bit FDP FIFO Descriptor Pointer to internal multi-user RAM is placed in address 0x4 of RCT. 11 This RCT is an MRCT and is used for Multicast shaping. See Section 39.14, "Multicast."
	2	**FE_AUTO_ CLR**	For Multicast using WFT algorithm only. 0 Auto FE clear receive mechanism is disabled. 1 Auto FE clear receive mechanism is enabled.
	3	-	Reserved. Initialized to 0.
	4	-	Reserved. Initialized to 0.
0x02	5-12	**MCGID**	Multicast source Group ID - indicate one of 256 available m/c source groups. Upon an event (MCE) the status of the m/c members that caused this event will be replicated in MCE_BASE + MCGID*4 in multi-user RAM. Also for Multicast FIFO using auto FE mechanism, indicate the pointer to the Pointer Table for per MCGID WFT Pointer Tables location in FEPRP_BASE+ MCGID*2. see Section 39.14.3.4, "Auto FE mechanism for FIFO Multicast Mode"
	13-15	**AAL**	AAL type: 000 AAL0 Reassembly with no adaptation layer. 001 AAL1 ATM adaptation layer 1 protocol. 010 AAL5 ATM adaptation layer 5 protocol. 011 AAL3. 100 AAl2. 101 CES. 110 MSP-Multi ServIce Protocol - switched ATM cells.
0x04	—	**Receive Cell Serial Number**	Receive Cell Serial Number. is SN assigned for each received cell by the receiver. All destination group member should transmit the cells in the order of this numbering. Initialized by the user.
0x08	—	**Cell Time Stamp**	Whenever a cell is received, the PowerQUICC II Pro MSP timestamp timer is sampled and written to this field. See Section 20.3.9, "QUICC Engine Time-Stamp Control Register (CETSCR)."
0x0C	—	**QDCC**	Queue Dropped Cell Count. Cells dropped due to policer.
0x0E	—	**CLP1 Count**	CLP1 cells Count.
0x10	—	**CLP0 Count**	CLP0 cells count.
0x14		**Next Receive Cell Pointer**	A pointer to the next cell in the linked list queue. Initialized by the user.
0x18	—	**UPCPTR**	Pointer to external UPC Parameter Table (UPCT). Aligned to 32 byte addresses.
0x1A		**Multicast Members Mask**	Holds a mask where each bit represents a member of the destination channels of the Tx. For example 0x00000005 represents destination members #31 and # 29 as represented in the MTCT[MCIDN]

Table 39-30. Multicast RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1E	0	**MS**	Match Status. Valid only for CC in UDH mode. 0 Match was found. 1 Match was not found. Cell will be discarded without updating the SRCT in external memory. The CPU can use this mechanism when performing dynamic changes in the SRCT/MRCT, without the interfering of the QUICC Engine block.
	1	**NCCLPIE**	Non-Conforming CLP1 or CLP0 counters overflow interrupt enable
	2-7	**PMT**	Performance monitoring Table Index. One of 64 performance monitoring parameter tables is chosen fir this channel
	8	**SEGT**	Segment OAM termination (used for OAM termination) 0 Do not terminate segment OAM cells 1 Terminate segment OAM cells (OAM cells are sent to Raw Cell Queue)
	9	**ENDT**	End-to-end OAM termination (used for OAM termination) 0 Do not terminate end-to-end OAM cells 1 Terminate end-to-end OAM cells (OAM cells are sent to Raw Cell Queue)
	10-11	**-**	Reserved. Initialized to 0.
	12	**MCRIE**	Multicast receiver event Interrupt Enable
	13	**QDCCIE**	QDCC Counter overflow Interrupt Enable
	14	**CLPIE**	CLP0C Counter or CLP1C Counter overflow Interrupt Enable
	15	**PME**	Performance monitoring Enable 0 No performance monitoring o this channel 1 Performance monitoring is enabled on this channel

NOTE

For an active channel, the RISC uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes. The user can alter the interrupt enable bits or the multicast members field on the fly.

39.14.9 MTCT- Multicast TCT

TCTM=11, AAL=110

	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			VCON	INTQ	
Offset + 0x02	TCTM			MCIDN					—	ADDCH	FIFO MODE	—		AAL		
Offset + 0x04	MQPTR															
Offset + 0x06																
Offset + 0x08	Transmit Serial Number															
Offset + 0x0A																
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10	CPL0 Count															
Offset + 0x12																
Offset + 0x14	CLP1 Count															
Offset + 0x16	APCLC															
Offset + 0x18	TVPI												TVCI			
Offset + 0x1A	TVCI												—	CVP	CVC	CGFC
Offset + 0x1C	Res		PMT													
Offset + 0x1E	INS_CELL_ Flag												MCETIE	STPT	CLPIE	PME

Figure 39-49. Multicast Transmit Connection Table (MTCT)

The RISC updates only entries 0x00-0x17 (24 bytes), and 1 byte located in 0x1e; other bytes are read by the RISC but are not updated by it. The CPU controlling the connection can change relevant bits from entries 0x18-0x1F (not including 0x1E) on the fly, and the RISC will act upon the new value, next time the TCT is used. The CPU shouldn’t update the byte in location 0x1E, after setting the INS_CELL_En bit - until the RISC reset this bit again.

Next Table (Table 39-31) describes general MTCT fields.

Table 39-31. MTCT Field Descriptions

Offset	Bits	Name	Description
0x0	0-1		Reserved. Initialize to zero.
	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 19.1.8.2, "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 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.
	6	**DTB**	Data buffers bus (Source linked list). If Insert cell mode is enabled and the cell resides in external memory, this bit should reflect the bus location of the cell to be transmitted. 0 Data buffers reside on the coherent system bus. 1 Data buffers reside on the secondary bus.
	7	**BDB**	Interrupt queues bus. If Insert cell mode is enabled and the CC>255 this bit should reflect the bus location of the Insert Cell pointers Table see Section 39.15.3, "Insert Cell Mode." 0 I reside on the coherent system bus. 1 reside on the secondary bus.
	8	**AVCF**	Auto VC off. APC Mode- 0 - The RISC doesn't remove this connection automatically, upon overwritten condition (when the serial number of the transmitter is smaller the actual serial number in the cell's COV). Instead this connection continues to transmit regularly by adopting the serial number of the COV, which is done by the RISC. 1 - The RISC remove this connection automatically by setting STPT, upon overwritten condition. FIFO Mode - 0 - The same as for APC mode. 1 - The RISC doesn't remove automatically the Transmit connection upon overwritten condition. It is the CPU's responsibility to remove this connection from the Channel code list structure of the corresponding FIFO.
	9	**-**	Reserved. Set to zero.

Table 39-31. MTCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x00	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. 11 Reserved.
	12	**-**	Reserved. Set to zero.
	13	**VCON**	Virtual channel is on. Only valid for APC Mode scheduling. Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPS] (stop transmit), the RISC 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 RISC clears VCON.
	14-15	**INTQ**	Points to one of four interrupt queues.
0x02	0-1	**TCTM**	TCT Mode 00 This is a STCT used as a leaf node for APC shaping 01 This is a HTCT used for a root node of hierarchical shaping 10 This is a FTCT used for Fast FIFO shaping 11 This is a MTCT used for Multicast shaping
	2	**---**	Reserved. Initialized to zero.
	3-7	**MCIDN**	m/c ID number is a number representing the destination channel number (0-31). The RISC upon transmitting a cell resets the corresponding bit in the Multicast Members field in the COV area of the cell. For example MCIDN = 7 will reset bit 7 in the COV.
	8	**---**	Should be initialized to zero.
	9	**ADDCH**	Set by the core on adding a MTCT to the source linked list, only when MRCT[Receive Cell Serial Number] is not equal to zero. See Section 39.14.5, “On-Line Modification of the Multicast Destination Group” for more information.
	10	**FIFO MODE**	Multicast mode used for scheduling the transmitter- 0 - APC like 1- FIFO WFQ
	13-15	**AAL**	AAL type: 000 AAL0 Reassembly with no adaptation layer. 001 AAL1 ATM adaptation layer 1 protocol. 010 AAL5 ATM adaptation layer 5 protocol. 011 AAL3. 100 AAl2. 101 CES. 110 MSP-Multi ServIce Protocol - switched ATM cells.
0x04		**MQPTR**	Multicast queue pointer (32 bits): Points to the next cell to be transmitted from the multicast source queue. Initialized by the user.
0x08		**Transmit Serial no.**	Transmit serial number. Should be initialized to zero.

Table 39-31. MTCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x0C	0–7	**Rate Remainder**	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Initialize to 0. Valid only on APC mode operation.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot.Valid only on APC mode operation.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Valid only on APC mode operation.
0x10	—	**CLP0 Count**	CLP0 cell count. Initialize to 0.
0x14	—	**CLP1 Count**	CLP1 cell count. Initialize to 0.
0x16	—	**APCLC**	APC Linked Channel. Used by RISC. Initialize to zero.Valid only on APC mode operation.
0x18	0-11	**TVPI**	Transmit VPI. Replaces VPI of cell if CVP bit is set (including GFC bits if CGFC is set)
	12-15	**TVCI**	Most significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set
0x1A	0-11	**TVCI**	Least significant bits of Transmit VCI. Replaces VCI of cell if CVC bit is set
	12	**Reserved**	Reserved. Initialize to zero.
	13-15	**CVP/CVC/CGFC**	CVP/CVC/CGFC (address translation) 000 no translation 001 NOT valid 010 Change VCI Portion of ATM address 011 NOT valid 100 Change VPI Portion of ATM address (including GFC) 101 Change VPI Portion of ATM address (NOT including GFC) 110 Change VCI & VPI Portion of ATM address (including GFC) 111 Change VCI & VPI Portion of ATM address (NOT including GFC)
0x1C	0-1	**-**	Reserved. Set to zero
	2-7	**PMT**	Performance monitoring Table Index. One of 64 performance monitoring parameter tables is chosen for this channel
	8-15	**-**	Reserved. Set to zero

Table 39-31. MTCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1E	0	**INS_CELL_ Flag**	Insert Cell Flag. Initialized to 0. Set by the RISC when insert cell host command is performed by the user, for inserting a cell to this connection. Reset by the RISC on completion of the inserted cell operation. See Section 39.15.3, "Insert Cell Mode." The CPU can only read this status bit in order to verify that the inserted cell was actually transmitted for this channel. Only after this bit was cleared another insert cell host command for this channel could be issued.
	1-11	**-**	Reserved. Set to zero
	12	**MCETIE**	MCET interrupt enabled.
	13	**STPT**	Stop transmit. Initialize to 0. When the host sets this bit, the RISC deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. This bit should not be used in switch mode (other than debug purposes). Host command Stop Rx should be used instead (so remaining cells are transmitted). The RISC reads 32 bytes of TCT, but writes back only 24 bytes. Therefore this bit is not altered by the RISC. Valid only for APC mode of operation.
	14	**CLPIE**	CLP0C Counter or CLP1C Counter overflow Interrupt Enable
	15	**PME**	Performance monitoring Enable 0 No performance monitoring on this channel 1 Performance monitoring is enabled on this channel

39.15 OAM Support

39.15.1 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. In switch mode the receive raw cell queue is used for removing management cells from the regular cells flow to the host. When a management cell is sent to the receive raw cell queue, the RISC sets OAM in the RxBD. Cell Filtering

Cell filtering is done according to the programming in the cell's associated RCT which depends on the switch function (End Point, Segment Point, Transparent point).

As shown in Figure 39-50, the main cell stream of a certain PHY is divided to several cell streams (Several VPC's, VCC's or combination of both).

The filtering may be done at the F4 level (Filtering of cells from the VPC's cell stream) or at the F5 level (Filtering of cells from the VCC's cell stream).

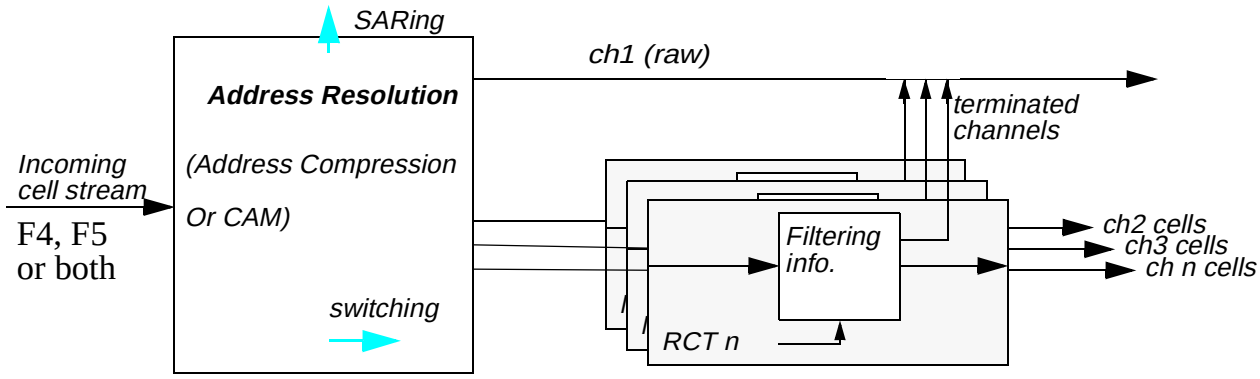


Figure 39-50. Management Cell Filtering

According to the cell's header and the required type of filtering, it will either be redirected to the raw cell queue (This action is called termination) or it will continue with the regular cell flow.

OAM cells are sent to the RAW Cell Queue if their associated RCT[SEGT], RCT[ENDT] and RCT[VPC] match the information extracted from the cell's header i.e whether it's an End Point OAM cell, Segment Point OAM cell and the switching level (F4 or F5).

Reserved VCI (6,7-15) are sent to the raw cell queue if the associated VCI bit in the VCI_Filtering entry in the parameter ram is set.

Following is a table that summarizes all the filtering options.

The first four entries are for VCC channels, the latter four are for VPC channels. Error cells are cells that should not have been received. The OAMERR bit is set in the extra header of those cells, and they are forwarded to the CPU.

Table 39-32. Cell Filtering Options

RCT[VPC]	RCT[SEGT]	RCT[ENDT]	F4 Segment (VCI=3)	F4 End-toEnd (VCI=4)	F5 Segment (PTI=100)	F5 End-toEnd (PTI=101)
0	0	0	Forward to CPU (OAMERR)	Forward to CPU (OAMERR)	Forward to Switch	Forward to Switch
0	0	1	Forward to CPU (OAMERR)	Forward to CPU (OAMERR)	Forward to CPU (OAMERR)	Forward to CPU
0	1	0	Forward to CPU (OAMERR)	Forward to CPU (OAMERR)	Forward to CPU	Forward to Switch
0	1	1	Forward to CPU (OAMERR)	Forward to CPU (OAMERR)	Forward to CPU	Forward to CPU
1	0	0	Forward to Switch	Forward to Switch	Forward to Switch	Forward to Switch

Table 39-32. Cell Filtering Options (continued)

RCT[VPC]	RCT[SEGT]	RCT[ENDT]	F4 Segment (VCI=3)	F4 End-toEnd (VCI=4)	F5 Segment (PTI=100)	F5 End-toEnd (PTI=101)
1	0	1	Forward to CPU (OAMERR)	Forward to CPU	Forward to Switch	Forward to Switch
1	1	0	Forward to CPU	Forward to Switch	Forward to Switch	Forward to Switch
1	1	1	Forward to CPU	Forward to CPU	Forward to Switch	Forward to Switch

1. Cells marked with OAMERR in the extra header are sent to the CPU. Those cells should not have been received in the first place

39.15.2 Performance monitoring (PM)

See Section 32.2.4, “Performance Monitoring.”

39.15.3 Insert Cell Mode

The user may require OAM cells to run in-band, i.e., shaped like the data stream they are part of. MSP features an “Insert Cell” mode, which enables the host to insert cells into APC or FIFO shaped channels on transmission.

An Insert Cell pointers Table was defined, see Figure 39-51. This Table contains one entry for each Channel Code in the system, and is located in internal RAM for the first 256 channels, or in external RAM, for channels codes which are bigger than or equal to 256. INT_INS_CELL_BASE and EXT_INS_CELL_BASE parameters were defined in the MSP Extended Parameter RAM. The user should always allocate at least one entry at INT_INS_CELL_BASE (associated with channel 0). This entry size should match the size as determined by the INS_CELL_Header_LOC parameter in the MSP Extended Page. Each Table entry consists of four bytes length pointer to the Insert Cell Template, and optionally another four bytes for the new cell header which will replace the existing cell header in the cell template. This could be useful for saving memory resources, and avoiding duplication of cells in memory when the difference between the Insert Cell templates of different VCs is limited to the cell header itself. The length of this entry is determined by the INS_CELL_Header_LOC parameter in the MSP Extended Page.

The Insert Cell pointer, or the cell template, should be aligned to 64 bytes, and could be an internal RAM address. Since there is a requirement for alignment of the cell address in memory, the 6LSB are available for usage. Setting the LSB to 1 implies [INS_CELL_LOC]=1 and therefore the cell address is on the external memory, if [INS_CELL_LOC] is cleared the cell address is in multi-user RAM. Setting the bit [CR10] in this entry means that CRC10 insertion is enabled. See Figure 39-52” for description. The size of the cell template is 52 bytes (HEC isn’t included), unless stated that a UDH mode is on, by the TUDC bit and TEHS field in the FPSMR register.

The user should prepare the cell including user-defined cell header (if exist) in a continuous manner so that the UDC comes first containing 12 bytes (no matter what is the number indicated in the FPSMR[TEHS] parameter), then the cell header containing 4 bytes (this header should be located even if INS_CELL_Header_LOC parameter is enabled), and the payload 48 bytes. If UDC is not enabled, then

the Cell header comes immediately at the beginning of the template. See Figure 39-51 for more details. Note that if the cell template is internal, then all the cell fields, i.e. UDH (if exist), header and payload, should be placed byte swapped (each 4 bytes should be swapped) to assure proper transmission. If the template is external, then the cell is located normally in Big Endian manner.

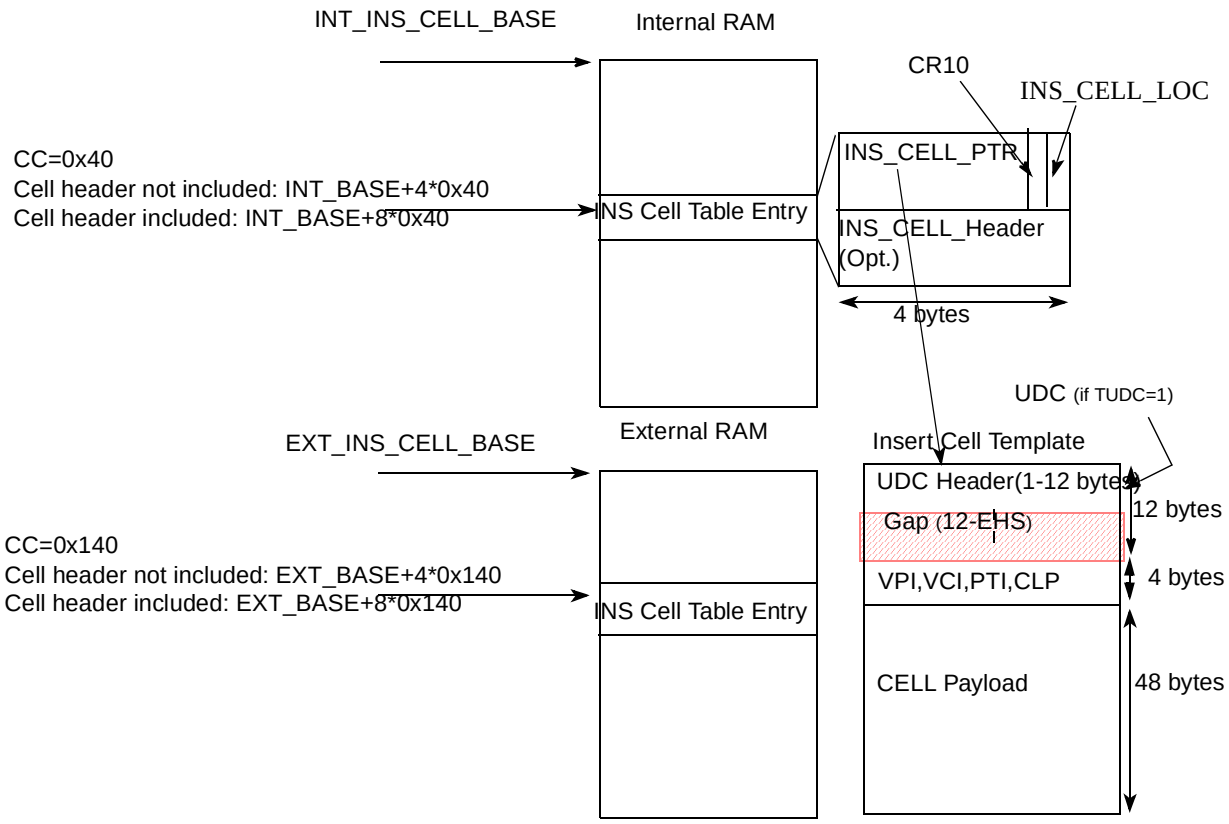


Figure 39-51. Insert Cell Table Description

The user inserts cell into a specific channel code using an host command, see Section 39.20.2, "MSP Insert Cell Command." The RISC set the INS_CELL_Flag bit in the corresponding xTCT(STCT, FTCT or MTCT), and also perform additional tasks in order to validate that the channel is valid in the APC table or in the FIFO. After transmitting the cell, the RISC clears this bit, so the user could resend another cell for this channel.

The user should monitor the INS_CELL_Flag bit after issuing the insert cell command, and only after it is cleared, a new insert cell command for this channel could be issued.

Notice that the xTCT should already be assigned to a specific SRCT or MRCT, as for normal MSP switch operation. The host command verifies that for STCT the channel will be rescheduled into the APC (since the AVCON/AVCOF mechanism), and for FTCT the channel would be entered into the FIFO list as there was a "dummy" receive operation.

Figure 39-52 shows the layout of the INS_CELL_PTR entry.

Field	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	Address (high)															
02	Address (Low)														**CR10**	**INS_CELL_LOC**
04	**Optional** Cell Header (VPI,VCI,PTI,CLP)															
06																

Figure 39-52. INS_CELL_PTR Entry

Table 39-33. INS_CELL_PTR Entry Description

Offset	Name	bits	Description
0x00	Address (High)	0-15	Inserted cell template address high.
0x02	Address (Low)	0-13	Inserted cell template address low.
	CR10	14	CRC-10 0 - CRC10 insertion is disabled. 1 - CRC10 insertion is enabled.
	INS_CELL_LOC	15	Inserted cell location 0 - Cell resides in multi-user RAM. Address high should be cleared. 1 - Cell resided in external memory.
0x04	Cell Header	Word	Cell header. Optional entry that depends on INS_CELL_Header_LOC parameter in the MSP Extended Page.

39.16 Global Mode Entry (GMODE)

See Global mode entry in Section 32.3.2.5, “Global Mode Entry (GMODE).”

39.17 ATM Parameter RAM Map

See more description in Table 32-18, “Sub-page0 Configuration Table.”

39.18 MSP Extended parameter RAM page

See description in Table , “Subpage1 Configuration Table”

39.18.1 FIFO Status Entry

FIFO Status Entry is shown in Figure 39-53.

Bit	0	1	2	3	4	5	6	7
Field	FS0	FS1	FS2	FS3	FS4	FS5	FS6	FS7

Figure 39-53. FIFO Status (FS) Entry

Table 39-34 describes the operation FIFO status bits.

Table 39-34. FIFO Status Entry

Bits	Name	Description
0-7	**FSi**	FIFO Status. Bit 7 for FIFO 7, bit 6 for FIFO6 ... bit 0 for FIFO0. For each FIFO, if its respective FS bit is set, the FIFO has cells queued in it; if the bit is reset, the FIFO is empty. If all bits in FS are reset, this HTCT has no cells in its lower hierarchy to be transmitted. Initialize to zero. The CPU does not access these bits during the operation of the MSP.

Each hierarchical connection has a FIFO Status entry associated with it. The location of this entry is located FSOff from FS_BASE in the multi-user RAM (FSOff is specified in the HTCT and in SRCT). The RISC updates the FS Status Entry.

39.18.2 FIFO Descriptor Pointers Table

The FIFO descriptor Pointers Table contains up to 128 entries used only for each PHY configured to FIFO mode (GMODE[MSP]=1 and Scheduler Parameter Table[AFM]=0). The FDT Pointer Table base address is pointed by the FDTP_BASE in the MSP parameter ram. The first entry is associated with PHY0, and the last entry with PHY 128. In this mode, the scheduling is done with WFQ. Each entry is 8 bytes long. A space of 1024 bytes is allocated for this table.

Only PHYs configured for FIFO mode (multicast or unicast) in the transmitter UCC side, have a valid entry in their respective location in this table.

Table 39-35 is described below:

Table 39-35. FIFO Descriptor Pointer Table

Address	Name	Width	Description
FDTP_BASE+ 0x00	FDTPTR0	Hword	Pointer to the first FIFO Descriptor Table entry (FDT) for PHY0.
FDTP_BASE+0x02	WFTMIN0	Hword	Used by RISC initialize to zero.
FDTP_BASE+0x04	res		
FDTP_BASE+0x06	APC_PARAM_PTR0		(APCP_BASE + 128*phy #0) This is the address of the Scheduler parameter table designated for FIFO mode to which this FDP correspond.
FDTP_BASE+ 0x00+8*i	FDTPTRi	Hword	Pointer to first FIFO Descriptor Table entry (FDT) for PHYi.

Table 39-35. FIFO Descriptor Pointer Table (continued)

Address	Name	Width	Description
FDTP_BASE+0x02+8*i	WFTMINi	Hword	Used by RISC initialize to zero.
FDTP_BASE+0x04+8*i	res		
FDTP_BASE+0x06+8*i	APC_PARAM_PTRi[1]		(APCP_BASE + 128*phy #i) This is the address of the Scheduler parameter table designated for FIFO mode to which this FDP correspond.
FDTP_BASE+ 0x3F8	FDTPTR127	Hword	Pointer to first FIFO Descriptor Table entry (FDT) for PHY128.
FDTP_BASE+0x3fa	WFTMIN127	Hword	Used by RISC initialize to zero.
FDTP_BASE+0x3fc	res		
FDTP_BASE+0x3fe	APC_PARAM_PTR127		(APCP_BASE + 128*phy #31) This is the address of the Scheduler parameter table designated for FIFO mode to which this FDP correspond.

1 "i" refers to PHY number

39.19 MSP Exceptions

The MSP interrupt handling involves two principal data structures, UCCEs (UCC event registers) and circular interrupt queues.

Four priority interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ], the user determines which queue receives the interrupt.

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 GINT, the host starts processing the interrupt queue and clears this entry's valid bit. See Section 39.19.2, "Interrupt Queue Entry in MSP mode." The host follows this procedure until it reaches an entry with V = 0.

The host controls the number of interrupts send to the CPU by programming the interrupt queues counters. When an event is sent to an interrupt queue, a counter is decremented. When a counter reaches zero, an UCCE[GINT*x*] is set. The counter should be user-initialized to INT_ICNT.

39.19.1 Interrupt Queues

See Interrupt queues description in Section 32.3.13.2, "Interrupt Queues."

39.19.2 Interrupt Queue Entry in MSP mode

Figure 39-54 shows an interrupt queue entry.

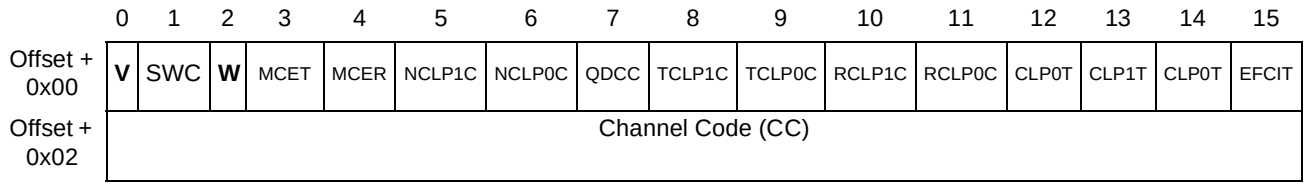


Figure 39-54. Interrupt Queue Entry

Table 39-36 describes interrupt queue entry fields if SWC=1.

Table 39-36. 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 RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	SWC	0 This interrupt queue entry is associated with SAR interrupt. 1 This interrupt queue entry is associated with Switch interrupt.
	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	MCET	Multicast Transmitter Error. Multicast destination Channel serial number does not match. The current cell has been overwritten by the receiver.
	4	MCER	Multicast Receive Error. Receive Channel has overwritten one or more Tx destinations channel cell. Also for FIFO Multicast mode, this bit together with EFCI bit indicates FIFO Disable interrupt.
	5	NCLP1C	NCLP1 Policing Counter Overflow
	6	NCLP0C	NCLP0 Policing Counter Overflow
	7	QDCC	Queue Dropped Cell Counter Overflow
	8	TCLP1C	Transmitter CLP1 Billing Counter Overflow
	9	TCLP0C	Transmitter CLP0 Billing Counter Overflow.
	10	RCLP1C	Receiver CLP1 Billing Counter Overflow
	11	RCLP0C	Receiver CLP0 Billing Counter Overflow.
	12	CLP01T	CLP01 Threshold violation. Bit is set when a cell is dropped due to Threshold violation.
	13	CLP1T	CLP1 threshold violation. Bit is set when a cell is dropped due to Threshold violation.
	14	CLP0T	CLP0 threshold violation. Bit is set when a cell is dropped due to Threshold violation.
	15	EFCI T	EFCI threshold violation. Bit is set when a the PTI bit in a cell was set due to crossing of this threshold. Also for FIFO Multicast mode, this bit together with MCER bit indicates FIFO Disable interrupt.

39.19.3 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 RISC clears the internal interrupt request.

UCCM is the ATM controller mask register. It is a 16-bit read/write register with the same bit format as UCCE. If an UCCM bit is set, the corresponding interrupt is enabled in UCCE. If it is cleared, the corresponding interrupt is masked. UCCM is cleared at reset.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—				FCPI	TIRU	GRLI	GBPB	GINT3	GINT2	GINT1	GINT0	INTO3	INTO2	INTO1	INTO0
Reset	0000_0000_0000_0000															
R/W	R/W															
Address	0x11310 (UCCE1), 0x11330 (UCCE2), 0x11350 (UCCE3)/ 0x11314 (UCCM1), 0x11334 (UCCM2), 0x11354 (UCCM3)															

Figure 39-55. ATM Event Register (UCCE)/UCC Mask Register (UCCM)

Table 39-37 describes UCCE fields.

Table 39-37. UCCE/UCCM Field Descriptions

Bit	Name	Description
0–3	—	Reserved, should be cleared.
4	FCPI	Free Cell Pool Interrupt. An interrupt related to the free cell pool has occurred.
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	SAR events
7	GBPB	SAR events
8–11	GINT*x*	Global interrupt. Set when an event is sent to the corresponding interrupt queue.
12–15	INTO*x*	Interrupt queue overflow. Set when an overflow condition occurs in the corresponding interrupt queue. This occurs when the RISC attempts to overwrite a valid interrupt entry.

39.20 MSP Host Commands

39.20.1 MSP FCP Commands

See Section 20.3.1, “QUICC Engine Command Register (CECR) for CECR programming description CECR to MSP host commands.

Before issuing the FCP (GET/PUT) command, the user should initialize COMM_INFO fields in the subpage0 configuration parameter RAM as described in Figure 39-56.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x12	—					FCPN	—									BIB
0x14	FCP_ADDRESS															
0x16																

Figure 39-56. COMM_INFO Field

Table 39-38 describes COMM_INFO fields

Table 39-38. COMM_INFO Field Descriptions

Offset*	Bit	Name	Description
0x12	0–4	-	n/a
	5	FCPN	Free Cell Pool number. Used only for Free Cell Pool Host commands. 0 - Free Cell Pool 1 1 - Free Cell Pool 2 * This FCP value must be the same as the FCP value in the TCT or RCT that related to the host command.
	6–14	—	n/a
	15	BIB	Interrupt queues and UPCT bus 0 reside on the coherent system bus. 1 reside on the secondary bus. *This Field must be the same as the BDB field in The RCT /TCT that related to the Host command.
0x14 - 0x16	0–31	FCP_ADDRESS	FCP address - The 32 bits word of the returned cell pointer during CP_GET command or the 32 bits word address given by the host to be returned to the selected FCP.

Note: *Offset from Global ATM Parameters Table pointer (See Table 32-18)

Table specifies the commands supported by the MSP:

Table 39-39. Command Code Description

Command Code value	Command Name	Description
0x00	**Start TX**	This command turns a passive channel into an active channel by scheduling it into the APC table. This command should only be used for VCs that are SARed. This command should not be used during normal operation of a switched VC (SRCT-STCT) with automatic scheduling where the transmitter is automatically activated by the switch when a cell for a given VC is received. This command is still applicable to other APC modes: - Unicast mode (SRCT-STCT) with non-auto scheduling. - Hierarchical mode (HTCT). - Multicast mode (MRCT-MTCT).
Performed by the CPU	**Stop TX**	Deactivated channel, by not rescheduling the channel in the APC. Performed by the RISC.U by setting STPT bit in TCT.
Performed by the CPU	**Stop RX**	Performed by the CPU by resetting the Match-Successful (MS) bit in the address look-up table (either in the CAM or in the address compression table) of the receive channel. This command is useful in case there is a need to deactivate one receive channel, out of many channels aggregated into one transmit channel. The transmitter sends the remaining cells, and (assuming the APC is used) once they are all transmitted, the APC does not reschedule them (either because the STPT bit in the TCT is set by the CPU, or because the AVCF bit has been set at initialization). If there is no aggregation of multiple receive channels into one transmit channel, issuing a Stop RX command eventually results in halting switching of cells from channel-code from both the receive and the transmit This method also ensures that no cells with the transmit channel code remain in the memory.
0x01	**Insert Cell**	Insert Cell command. The insert cell command set the corresponding INS_CELL_Flag in the xTCT. This command also make sure that the corresponding xTCT will be valid in the APC table or in the FIFO, depending on the mode selected. For STCT mode which is using AVCON/AVCOFF mechanism, the activation of a channel in the APC is done by checking the TL=0, QCC=0 field in the STCT. For fifo mode, the FTCT is entered into the FIFO as there was a “dummy” receive operation of the SRCT that is normally used to be switched to this specific FTCT. The user should monitor the INS_CELL_Flag bit after issuing the insert cell command, and only after it is cleared, a new insert cell command for this channel could be issued.
0x02	**CP_GET**	Cell Pool GET command fetches a cell pointer out of the pool (The Pool number determined by COMM_INFO[FCPN] bit) and return it’s 32 bit address in COMM_INFO[FCP_ADDRESS_H] and COMM_INFO[FCP_ADDRESS_L] fields. The User will use the returned address to configure the relevant parameters according to the MSP mode of operation.
0x03	**CP_PUT**	Cell Pool PUT command puts the cell pointer which it’s specified in the COMM_INFO[FCP_ADDRESS_H] and COMM_INFO[FCP_ADDRESS_L back to the specified Pool (The Pool number determined by COMM_INFO[FCPN] bit). The user will use this command to return a cell to a Pool after the ATM channel is deactivated.

NOTE: Entries in boldface must be initialized by the user.

39.20.2 MSP Insert Cell Command

The Insert cell command is issued just like UCC ATM transmit command, see Section 39.20.1, “MSP FCP Commands,” with the CECR[SCB[page]] field set to the page in which the desired xTCT (the xTCT that belongs to the inserted cell) resides in.

39.20.2.1 Insert Cell in STCT Mode

The COMM_INFO fields for STCT mode should be initialized in the parameter RAM as described in Figure 39-57.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x12	-						GCRA Sched_ AVCON	AVCON						MSP MODE		
0x14	Insert Cell Channel Code															
0x16	TAPTR/ GCRA Scheduler PHY Parameter Table ptr															

Figure 39-57. COMM_INFO Field for STCT Mode

Table 39-40 describes COMM_INFO fields for STCT

Table 39-40. COMM_INFO Field Descriptions for STCT Mode

Offset	Bit	Name	Description
0x12	0–5	-	n/a
	6	GCRA Sched_AVCON	DRR APC AVCON. Valid only if AVCON bit is set. Indicates which type of APC is used to activate the channel. 0 - Traditional APC. 1 - GCRA Scheduler.
	7	AVCON	Auto VC ON (should be copied from the corresponding SRCT). If set the STCT is scheduled into the APC if STCT[QCC]=0 and STCT[TL]=0.
	13-15	MSP Mode	MSP Mode 000 - STCT 001 - Hierarchical FIFO 010 - WFQ FIFO 011 - MCST APC 100 - MCST FIFO (Non auto FE) 101 - MCST FIFO Auto FE mode 110 - res 111 - res
0x14	—	InsertCell Channel Code	Transmitter channel code of the insert cell.

Table 39-40. COMM_INFO Field Descriptions for STCT Mode (continued)

Offset	Bit	Name	Description
0x16	0-15	**TAPTR**	Valid only if AVCON mode is enabled. **Traditional APC mode:** TAPTR: Transmitter APC Pointer. This entry is the pointer to the APC priority Table entry in multi-user RAM. This is used by the RISC to automatically add a connection to the APC (if it was previously removed since its queue was empty). **GCRA Scheduler mode:** 14 most significant bits of the pointer to the GCRA Scheduler PHY Parameter Table.
	0-13	**GCRA Scheduler PHY Parameter Table ptr**	
	14-15	**PRI**	GCRA Scheduler Priority.

NOTE: Entries in boldface must be initialized by the user.

39.20.2.2 Insert Cell in Hierarchical and WFQ FIFO Modes

The COMM_INFO fields for Hierarchical and WFQ FIFO modes should be initialized in the parameter RAM as described in Figure 39-58.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x12	**-**													**MSP MODE**		
0x14	**RCT CC**															
0x16	**SRCT_TMP**															

Figure 39-58. COMM_INFO Field for FTCT/HTCT Mode

Notice that the user should allocate additional 8 bytes in Ram, see INS_CELL_RCELL_TMP parameter in Section 39.18, “MSP Extended parameter RAM page.” This parameter should be defined in the same page as CECR[SCB[page]] host command field indicates. Also notice that for both Hierarchical and WFQ FIFO modes the INS_CELL_Flag bit is set in the FTCT.

NOTE

The insert cell command for Hierarchical mode is valid for both modes of operations: HTCT AVCON/AVCF mode and Non HTCT AVCON/AVCF mode.

Table 39-41 describes COMM_INFO fields for STCT

Table 39-41. COMM_INFO Field Descriptions for STCT Mode

Offset	Bit	Name	Description
0x12	0-12	-	res.
	13-15	**MSP Mode**	MSP Mode 000 - STCT 001 - Hierarchical FIFO 010 - WFQ FIFO 011 - MCST APC 100 - MCST FIFO (Non auto FE) 101 - MCST FIFO Auto FE mode 110 - res 111 - res
0x14	—	**RCT CC**	RCT Channel Code. Consist the Receiver Channel code that is normally switched to the FTCT in which we want to insert the cell.
0x16	-	**SRCT_TMP**	SRCT temporary pointer. Allocates 32 bytes in RAM (should be 32 bytes aligned). This area should be Initialized to 0.

NOTE: Entries in boldface must be initialized by the user.

39.20.2.3 Insert Cell in MCST Auto FE Mode

The COMM_INFO fields for STCT mode should be initialized in the parameter RAM as described in Figure 39-59.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x12	-					-		-						**MSP MODE**		
0x14	**Insert Cell Channel Code**															
0x16	**WFT_PTR**															

Figure 39-59. COMM_INFO Field for MCST Auto FE Mode

Table 39-42 describes COMM_INFO fields for MCST Auto FE mode

Table 39-42. COMM_INFO Field Descriptions for MCST Auto FE Mode

Offset	Bit	Name	Description
0x12	0–4	-	n/a
	5-12	-	res.

Table 39-42. COMM_INFO Field Descriptions for MCST Auto FE Mode (continued)

Offset	Bit	Name	Description
	13-15	**MSP Mode**	MSP Mode 000 - STCT 001 - Hierarchical FIFO 010 - WFQ FIFO 011 - MCST APC 100 - MCST FIFO (Non auto FE) 101 - MCST FIFO Auto FE mode 110 - res 111 - res
0x14	—	**InsertCell Channel Code**	Transmitter channel code of the inserted cell.
0x16	**-**	**WFT_PTR**	WFT PTR. Pointer to the WFT entry which belongs to the FIFO where the MTCT resides. This parameter is used by the RISC in order to clear the FE bit in the WFT entry on event of host command.

NOTE: Entries in boldface must be initialized by the user.

39.20.2.4 Insert Cell in MCST APC/MCST FIFO Non Auto FE Modes

The COMM_INFO fields for MCST APC/MCST FIFO Non Auto FE modes should be initialized in the parameter RAM as described in Figure 39-60.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x12	**-**					**-**		**-**						**MSP MODE**		
0x14	**Insert Cell Channel Code**															
0x16	**-**															

Figure 39-60. COMM_INFO Field for MCST APC/MCST FIFO Non Auto FE Modes

Table 39-43 describes COMM_INFO fields for MCST APC/MCST FIFO Non Auto FE modes.

Table 39-43. COMM_INFO Field Descriptions for MCST APC/MCST FIFO Non Auto FE Modes

Offset	Bit	Name	Description
0x12	0–4	**-**	res.
	5-12	**-**	res.
	13-15	**MSP Mode**	MSP Mode 000 - STCT 001 - Hierarchical FIFO 010 - WFQ FIFO 011 - MCST APC 100 - MCST FIFO (Non auto FE) 101 - MCST FIFO Auto FE mode 110 - res 111 - res

Table 39-43. COMM_INFO Field Descriptions for MCST APC/MCST FIFO Non Auto FE Modes

Offset	Bit	Name	Description
0x14	—	**InsertCell Channel Code**	Transmitter channel code of the inserted cell.
0x16	**-**	**-**	res.

NOTE: Entries in boldface must be initialized by the user.

Chapter 40
L2 Ethernet Switch

40.1 Overview

The L2 Ethernet switch in the QUICC Engine block offers up to eight connection ports of 10/100 Mbps with MII/RMII Ethernet external ports and one CPU internal port. In addition, the switch also provides VLAN functionality, IGMP snooping, store-and-forward switching operation, and packet-error filtering.

Each port in the switch supports 4 priority levels; the priority is determined either by the VLAN tag priority or the IP TOS field in the frame header. In this way, quality of service (QoS) is provided for different types of traffic, such as voice, video, and data.

The L2 microcode-based Ethernet switch in the QUICC Engine block runs on top of the UCC when configured in Ethernet mode. It is possible to configure any number of UCCs (up to eight) as switch ports. Figure 40-1 depicts an example of a 8-port switch with eight UCCs dedicated to eight external switch ports and the ninth port as the CPU interface.

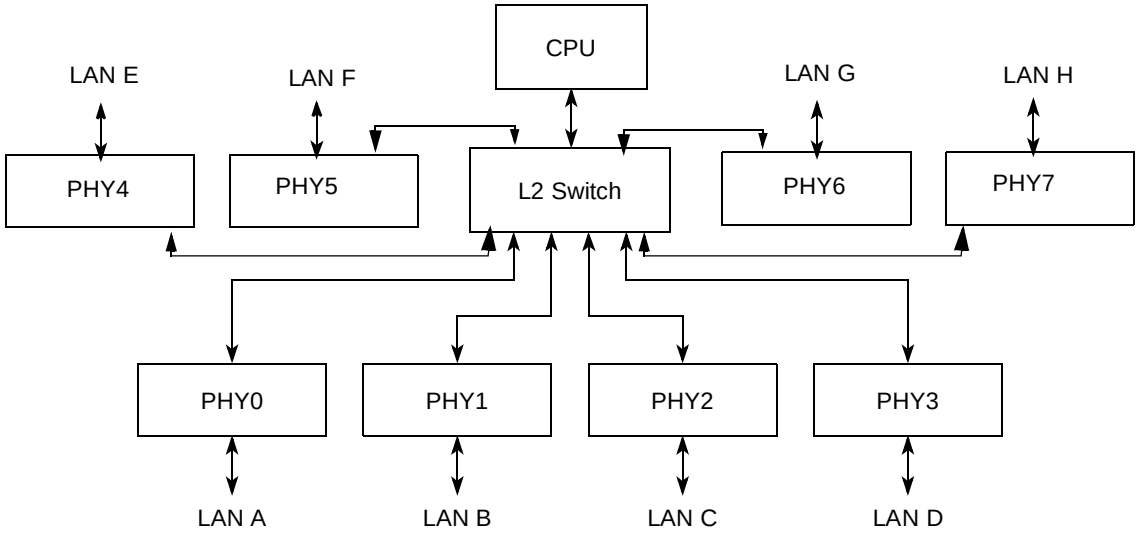


Figure 40-1. 8-Port Switch System

40.2 Features

The L2 switch supports the following features:

- L2 switch Ethernet controller
  - — Supports up to eight 10/100 Mbps full-/half-duplex Ethernet ports and one CPU internal port
  - — Independent MII/RMII interface for each port
  - — Serial management interface MDC/MDIO
- Internal and external memory
  - — External memory used as a packet storage buffer for up to 256 packets
  - — Multi-user RAM used for control structure and as a temporary packet storage buffer
  - — Size of multi-user RAM is programmable
  - — Internal address learning table (ADLT)
    - – Programmable table size
    - – Up to 2048 addresses using up to 16 Kbytes of multi-user RAM
    - – Self-learning mechanism
    - – Supports aging function
  - — Supports unicast/multicast/broadcast transmission
- VLAN
  - — Supports programmable number of virtual LANS (typically 64)
  - — Dynamic VLAN membership (per port or per VLAN ID)
  - — Tag insertion and extraction
  - — Supports tag-based priority (up to four priorities)
  - — Supports double VLAN tagging
- Flow control
  - — Supports flow control via IEEE 802.3 Pause packets for full duplex in case buffer is almost full
  - — Supports back pressure (collision mode) function for half duplex in case buffer is almost full
- Supports standard packet length for basic and VLAN modes. Programmable minimum and maximum frame length.
  - — Supports jumbo frames up to 9 Kbytes
- Statistical counters (RMON)
- Supports Ethernet II & 802.2 SNAP frames
- IGMP snooping
- QoS determined per port, tagged frame priority, IPv4’s type of service (TOS), IPv6’s traffic class (TC), or default priority
- Recognizes IEEE802.1D/802.1W/802.1S spanning tree protocol (BPDU packets) and forwards them to CPU
- Supports input port mirroring (for good frames)

40.3 Functional Overview

The following sections provide a functional overview of the L2 Ethernet switch in the QUICC Engine block.

40.3.1 Ethernet Ports

The L2 switch supports up to eight 10/100 BaseT Ethernet ports. These ports may be configured as half- or full-duplex. Each port includes a MAC, which is compliant with the IEEE 802.3 standard.

40.3.2 Frame Filtering

The L2 switch makes filtering decisions for each incoming packet based on its frame routing table, VLAN mapping, port state, and the system configuration. The switch may either forward the frame, if forwarding criteria are met, or discard it. All parameters are user-programmable. See Section 40.5, “Filtering Database,” for more details.

Incoming frames are discarded for the following situations:

- If an error such as a FCS error or short/long frame error is found in the incoming frame.
- If the destination port number is the same as the port on which the packet was received, unless the user enables this feature as described in Section 40.5.5, “Forwarding Frame Back to the Source Port.”
- If the source or destination port is disabled. (The port state can be changed according to the spanning tree protocol.)
- If the input buffer of the port is full. It is recommended that flow control be used to reduce the chance of packet loss.

40.3.3 Frame Forwarding

The switch forwards an incoming frame between its ports according to its destination address (DA). The switch uses a hash algorithm to learn the MAC address (source address, SA) and can learn up to 2K MAC addresses. The ADLT is stored in the internal multi-user RAM; the size of the table is programmable by the user. See Section 40.5.1, “Address Learning Table,” for more information.

The address lookup engine attempts to match the destination address with the addresses stored in the ADLT.

A unicast frame is forwarded to a destination port. If the destination port does not belong to the VLANs specified at the source port, the frame is discarded.

A multicast frame is forwarded to several destination ports according to the VLAN mapping and port state. See Section 40.5.3, “VLAN Destination Table (VDT),” for more details.

Frames with an unknown unicast or multicast destination are forwarded to all ports according to the default VLAN group.

40.3.4 Illegal Frames

The switch discards all the following illegal frames:

- Frames with a length smaller than the programmable minimum (typically 64 bytes),
- Frames with a length larger than the programmable maximum.
- Frames with a FCS error.
- Frame with line errors.

All of these frames are transmitted to the CPU in Error Debug mode.

40.3.5 FCS Checking

The FCS of received frames is checked for correctness. When a frame is transmitted, a new FCS is appended to account for any insertion or removal of a VLAN tag.

40.3.6 Address Aging

The address aging function supports the ability to automatically clear outdated entries in the ADLT. An aging timer is used to determine the aging time period. A learned source address (SA) is cleared if it is not updated by the address learning process during this aging time period. For security purposes, it is possible to disable the aging function on an entry basis.

40.3.7 IGMP Snooping

The L2 switch supports IGMP snooping. Without IGMP snooping, multicast traffic is forwarded to all ports, similar to broadcast traffic. With IGMP snooping, the multicast traffic of a group is only forwarded to ports that are members of that group. The IGMP snooping feature provides a significant reduction in multicast traffic through the switch.

40.3.8 Spanning Tree Protocol (BPDU Packets)

The switch recognizes bridge protocol data unit (BPDU) packets and sends them to the CPU for processing. The purpose of these packets is to provide path redundancy while preventing undesirable loops in the network. As a result of receiving BPDU packets, the CPU can disable ports to receive and transmit regular frame and/or BPDU packets.

40.3.9 Basic and Tag Modes

The L2 switch has the following two programmable modes of operation:

- Basic mode, which is used in a single LAN environment.
- Tagged mode, which is used for full VLAN support.

The switch supports a programmable number (typically 64) of different VLAN identifiers (VIDs) out of possible 4096 identifiers.

Each port of the switch can be assigned to one or multiple VIDs. Frames from the source port are forwarded to destination ports within the same VLAN domain. A broadcast/multicast frame is forwarded to all ports within the VIDs of the source port except the source port itself. A unicast frame is forwarded to the destination port only if the destination port is in the same VLAN as the source port. Otherwise, the frame is treated as a frame with an unknown DA. In basic mode, all ports belong to one undefined VID.

40.3.10 Flow Control

The L2 switch can operate at two different modes, half- and full duplex.

A pause frame is sent on a source port if that port exceeds a user-defined threshold value on the number of buffers currently used by that source port. Pause frames are transmitted on all ports if a user-defined global buffer usage threshold value is exceeded.

Back pressure is supported for half-duplex operation. When buffers of the switch are almost full, the switch transmits a JAM signal through the relevant ports. The remote station defers transmission after sensing the JAM signal.

In full-duplex mode, the switch transmits and receives frames in accordance to the 802.3x. In this mode, the transmission channel and the receiving channel operate independently. When buffers of the switch are almost full, the switch transmits a PAUSE frame with the maximum delay value. It also sends a new pause packet (with a pause period of 0x0000) to the transmitter to start sending new packets.

The transmission of a PAUSE frame reduces the chance of a packet loss.

40.3.11 Internal/External Memory

The L2 switch uses internal/external memory allocations for control structures (BDs, configuration parameters), packet buffering, the ADLT, and the VLAN table. The features of the internal/external memory allocations are listed as follows:

- The switch supports up to 256 packet buffers. The frame size is programmable. Jumbo frames are supported.
- The switch provides an on-chip ADLT with up to 2K entries for frame destination look-up.
- The switch provides an on-chip VID table and supports a programmable number (typically 64) of different VIDs out of a possible 4096 identifiers.

The overall multi-user RAM internal memory usage is programmable. See Section 40.9.1, “Internal Memory,” for details.

40.3.12 MII

The UCCs support the IEEE 802.3 media-independent interface (MII).

40.3.13 RMII

The UCCs provide a reduced media-independent interface (RMII) to the Ethernet MAC. The RMII interface is designed based on the standards and specifications that are documented in *Reduced MII Interface*, at AMD Inc., Broadcom Corp., National Semiconductor Corp., and Texas Instruments Inc.

The purpose of RMII is to provide a low-cost alternative to the IEEE802.3u MII interface (MII). Architecturally, the RMII is an additional reconciliation layer on either side of the MII. However, it can be implemented in the absence of a MII.

40.4 SWITCH Components

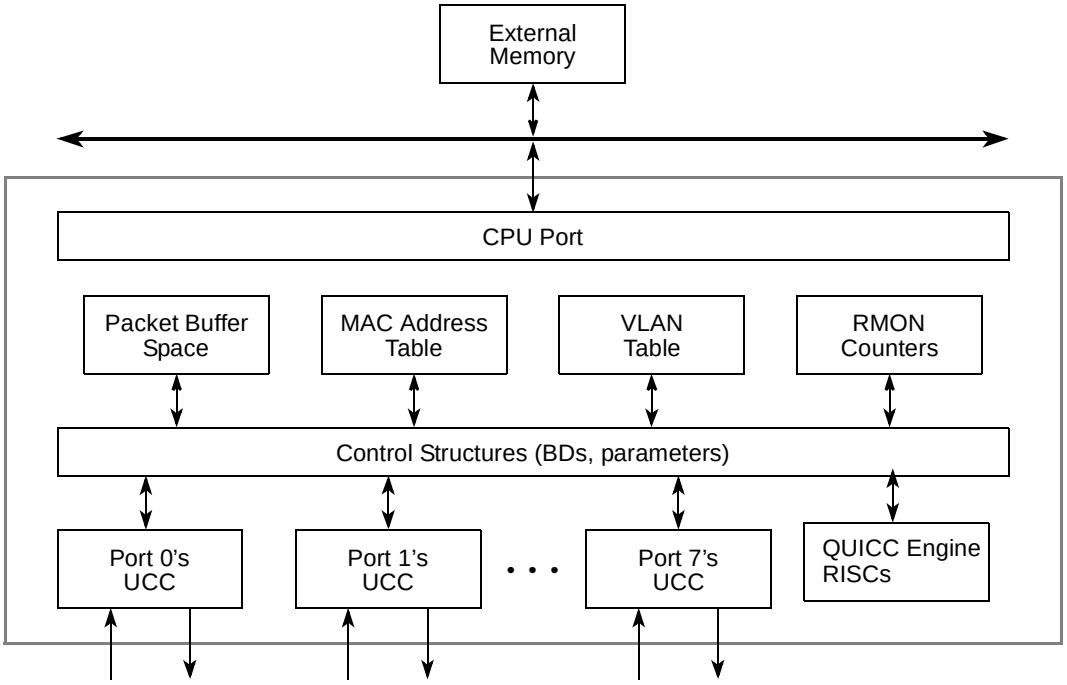


Figure 40-2. QUICC Engine Switch Components

40.4.1 Ingress Rule

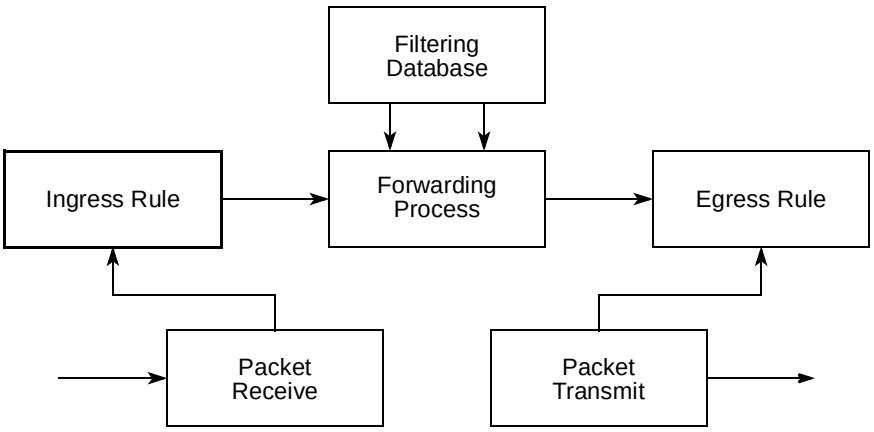


Figure 40-3. Switch Processing Flow

The purpose of the ingress rule is to define which frames are legal frames that can pass through the switch, and which are illegal and have to be either dropped or sent to the CPU for debugging.

The ingress rule is affected by the switch operation mode, e.g. basic or tag mode, and by the arriving frame, specifically the data or management frame.

Each port is capable of passing data or management (IGMP, BPDU, and flow control packets) tagged or untagged frames.

Figure 40-4 shows the Ethernet II / 802.3 / 802.2 SNAP frame formats.

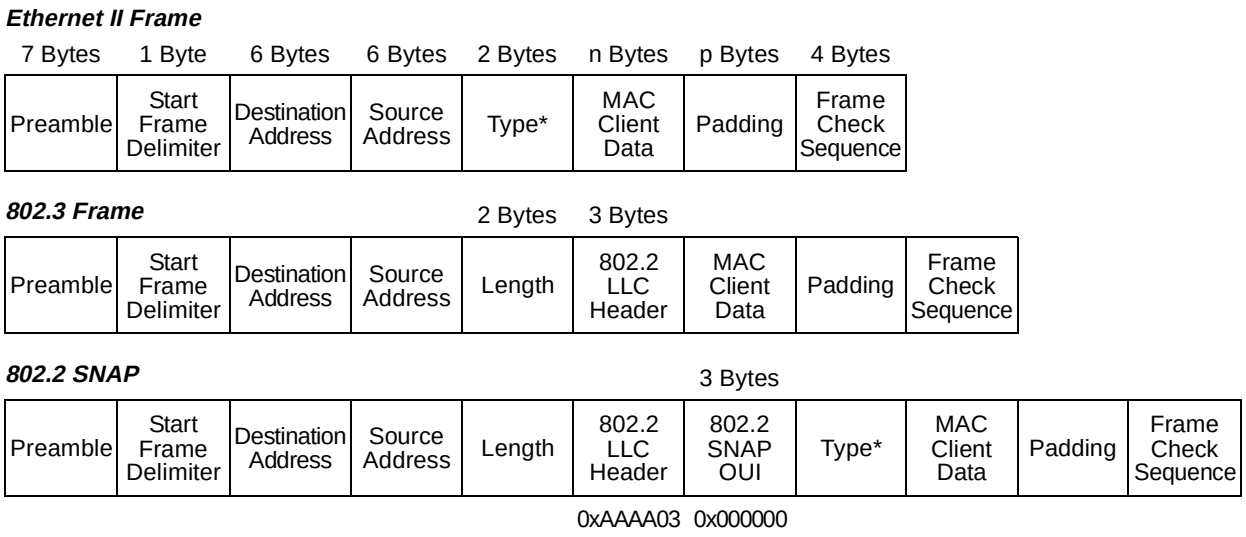


Figure 40-4. Ethernet II/802.3/802.2 SNAP Frame Formats

Figure 40-5 shows the basic and tag frame formats.

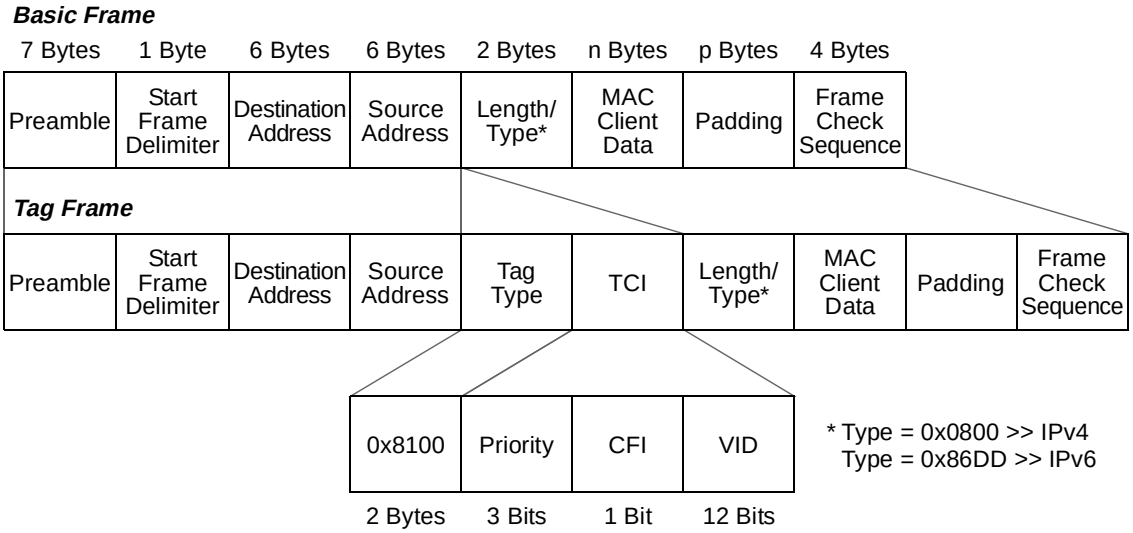


Figure 40-5. Basic/Tag Frame Formats

The L2 switch supports standard and jumbo frames with programmable length. See the MFLR field in Section 40.12.2, "Switch PRAM," for more details. Up to four priorities are supported out of a possible eight. The switch uses a priority table for mapping the original priority bits [2–0] to two priority bits. Section 40.13.3, "Priority Mapping Table (PMT)," provides more details.

40.4.2 IPv4 & IPv6 Frames (IP Multicast Support)

The L2 switch supports the IPv4 & IPv6 frame formats. It treats these frames as regular tagged/untagged frames with no priority, tag priority, or IP priority. IP priority includes a 6-bit priority field (TOS in IPv4 and TC in IPv6). The switch uses an IP priority mapping table to map the original priority bits [5–0] to two priority bits. See Section 40.13.4, "IP Priority Mapping Table (IPMT)," for more details. Table 40-1 shows the IPv4 & IPv6 header format.

Table 40-1. IPv4 & IPv6 Headers

Ver	IH	TOS	Total Length
Identification		Flags	Fragment
TTL		Protocol	Header Checksum
Source Address			
Destination Address			
Options			Padding

Ver	TC	Flow Label	
Payload Length		N Header	Hop Limit
Source Address			
Destination Address			

Field's name kept from IPv4 to IPv6
Fields not kept from/in IPv6
Name and position changed in IPv6
New field in IPv6

If IGMP snooping is enabled in the port, the switch assumes the header formats and can use them for multicast filtering. See Section 40.4.5, “IGMP Snooping” for more information.

40.4.3 Illegal Frames

The L2 switch either discards all illegal frames or sends them to the CPU for debugging. See the DM fields in Section 40.13.1, “Switch Parameters (SWPAR),” for details. Some examples of illegal frames are bad FCSs and short or long frames, smaller or larger than the programmable min/max frame lengths.

40.4.3.1 Tag Mode Ingress Rule

When an untagged frame is received on a port, the ingress process may drop the frame or add a default VID to it. See Section 40.7.2, “Tag Mode Switching,” for more information.

When a tagged frame is received on a port, the ingress process may deliver or drop the frame depending on the ingress VLAN destination table.

40.4.3.2 Management Frame

When an IGMP or BPDU frame arrives, the ingress process delivers it to the CPU for processing.

40.4.4 Default Tag

In tag mode, the L2 switch may add or replace the original tag with a default tag. Each port has its own default tag.

The default tag is used in the following situations:

- An untagged frame arrives
- A tagged frame arrives, and its VID is equal to zero
- A frame is in forced default tag mode

When a tag is inserted into an untagged frame, four bytes comprising the default tag are inserted immediately following the source address field. Refer to Figure 40-5 for the basic/tag frame format.

- The first byte is always 0x81.
- The second byte is always 0x00.
- The next three bits are the priority bits. They are copied from the default priority entry unless the frame is already tagged in which case its original priority bits remain.
- The next bit is always set to zero.
- The next twelve bits are the VID bits. These bits are copied from the default VID entry.

40.4.5 IGMP Snooping

The IGMP protocol is designed to specify membership of multicast groups. Using IGMP group memberships, the L2 switch can avoid flooding the network with multicast traffic by forwarding multicast packets only to specific ports according to which multicast group the packets are destined.

The L2 switch supports IGMP snooping. IGMP allows a host to inform the router that it wishes to receive packets addressed to a specific multicast group. These packets are always set to highest priority.

As the name implies, the switch snoops on the IGMP transmissions between the host and the router to keep track of multicast groups and member ports. A packet received on any of the eight ports is identified as an IGMP packet under all of the following situations:

- DA is multicast or broadcast.
- The type is IP (0x800 for IPv4 or 0x86DD for IPv6). See Figure 40-5 for the basic/tag frame format.
- {Protocol == IGMP (0x02)} / {Payload type (Next header) == 0x02}. See Table 40-1 for the IPv4 & IPv6 headers.

The switch-identified IGMP packet is delivered to the CPU for processing. The user can define more ports as destinations for these packets. See Section 40.13.1, "Switch Parameters (SWPAR)," for more information. However, these ports, including the CPU port, have to be members of the frame VID.

The CPU processes the IGMP frame and updates the GDA field details in the ADLT. See Section 40.5.1.4, "Group Destination Address (GDA)," for more information. This is done by using the command register; see Section 40.11.1, "Command Register," for details. The GDA field represents the actual port group for each multicast address.

40.4.6 Spanning Tree Protocol (STP)

The spanning tree protocol is a link-management protocol that provides path redundancy while preventing undesirable loops in the network. The IEEE 802.1D/802.1W/802.1S (STP/Rapid STP/Multiple STP), spanning tree is supported by the L2 switch using the CPU that runs this algorithm.

The example in Figure 40-6 shows a loop between three switches. When the spanning tree runs, one of the paths will be blocked automatically.

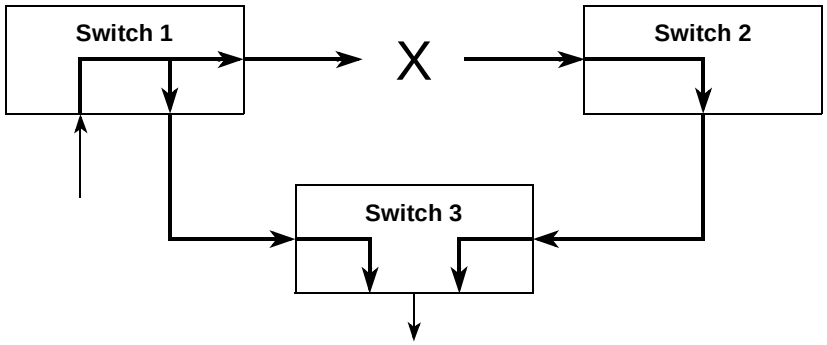


Figure 40-6. Example of Spanning Tree Preventing a Loop

The L2 switch supports the spanning tree protocol, which is recognized by the BPDU frames. It delivers these spanning tree frames to the CPU for processing and can change the state of the port (in case of STP/RSTP) or remove/add the port from/to any VLAN group (in case of MSTP). These packets are always set to highest priority.

A packet is recognized as spanning tree (BPDU frame) when the DA lies between 01-80-C2-00-00-00 and 01-80-C2-00-00-FF (with the exception of flow control pause packets, which use 01-80-C2-00-00-01).

The BPDU frame is delivered only to the CPU even though it is a multicast frame.

The BPDU indication, along with its source port, is supplied to the CPU by the Rx CPU buffer descriptor (RCBD). See Section 40.8.11.1, "Rx CPU BDs (RCBD)," for details.

As a result of these frames, the CPU may change the port's state. See Section 40.5.4, "Destination Port State Vector (DPSV)," and the PS field in Section 40.13.2, "Port Parameters (PPAR)," for more information.

40.4.6.1 Port State

The spanning tree protocol allows each port to work in one of the following states:

1. Forwarding — Regular operation (default mode). The port is open to all types of frames (incoming and outgoing). The ADLT is updated for any new good frame.
2. Disabled — The port is closed to all frames. The ADLT is not updated.
3. Blocking/Listening — The port is open (incoming and outgoing) only to BPDU frames. All other frames are dropped, and the ADLT is not updated.
4. Learning — The port is open (incoming and outgoing) only for BPDU frames. All other frames are dropped, but the ADLT is updated for new good frames.

NOTE

All of these states are relevant if spanning tree mode is enabled. If spanning tree mode is disabled, the L2 switch refers to the frames as regular multicast frames. See Section 40.13.1, "Switch Parameters (SWPAR)," for details.

40.4.7 Flow Control Packet

The switch recognizes the flow control packet (DA equal to 01-80-C2-00-00-01) and discards the packet after it is processed by the MAC control sublayer of the port. In Flow Control debug mode, the packet is sent to the CPU. See Section 40.3.10, "Flow Control," for more information.

40.4.8 Quality of Service (QoS)

The L2 switch supports up to four priority queues per port. The ingress rule can determine different priorities for each frame.

The following are the priority frame options:

- Error frames (in Error Debug mode), are always set to lowest priority.
- For a tagged frame with priority bits, the switch uses the priority table to map the original priority bits [2–0] to two priority bits. See Section 40.13.3, "Priority Mapping Table (PMT)," for details.
- An untagged frame gets a default priority per its source port. Each port has three default priority bits, and the switch treats them as if they have original frame priority. See Section 40.13.1, "Switch Parameters (SWPAR)," for more information.

- Unknown unicast destination addresses are always set to lowest priority.
- IGMP snooping and BPDU packets are always set to highest priority.
- The IPv4 and IPv6 packets have six priority bits (TOS[2–7] or TC[2–7]). The switch and the IP priority table map these six bits to two priority bits. See Section 40.13.4, “IP Priority Mapping Table (IPMT),” for more information.

NOTE

If the incoming frame is either tagged or is an IP frame, the user can choose which priority (IP or tag) needs to be used.

The lowest priority level is mapped from LOWEST_PR in Switch PRAM.

The highest priority level is mapped from the HIGHEST_PR in Switch PRAM.

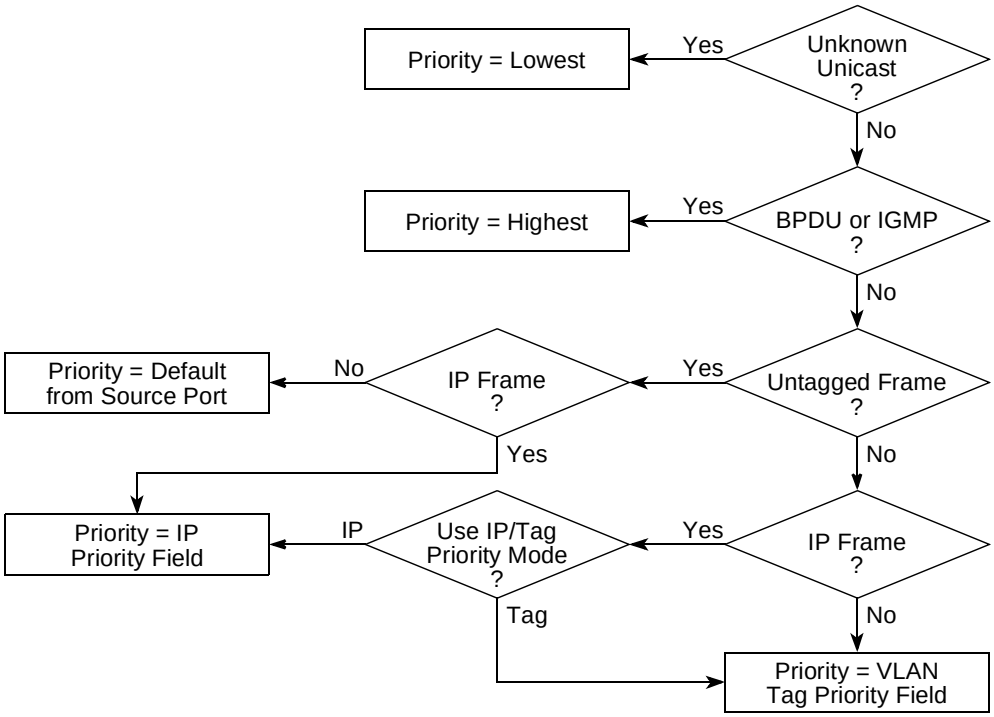


Figure 40-7. QoS Priority in Switch

40.4.9 Broadcast Storm Control

A broadcast storm occurs when message that has been broadcast across a network results in even more responses, and each response results in still more responses in a snowball effect. Errors in the network configuration can cause a storm.

In case of broadcast storm, the switch can block any port from receiving broadcast frames. The CPU can poll the RMON counters at intervals and if the rate of the broadcast traffic crosses a set threshold, disable the port receiving the broadcast. See BCD field in Section 40.13.2, “Port Parameters (PPAR).”

40.5 Filtering Database

The switch filtering database is an integral part of the switch. It is used by the switch to determine if a packet should be forwarded and which port it should be forwarded through.

The switch can apply multiple filtering databases when forwarding frames, with database choices programmed as part of the operating mode. The switch operation modes are as follows:

- Basic or tag mode. See Section 40.13.1, “Switch Parameters (SWPAR),” for information.
- IGMP snooping. See Section 40.13.2, “Port Parameters (PPAR),” for information.
- Spanning tree protocol. See Section 40.13.2, “Port Parameters (PPAR),” for information.

The switch filtering databases are as follows:

- Address learning table (ADLT) with Group destination address (GDA)
- Port designation table (PDT)
- VLAN destination table (VDT)
- Port state vector (PSV)

40.5.1 Address Learning Table

The address learning table (ADLT) learns the MAC source address of all arriving frames. Up to 2048 MAC addresses can be stored in the ADLT. See Table 40-13 for the internal memory usage for the ADLT. A minimum of four entries is required, and the number of entries must be a multiple of four. Each entry in the table is 64 bits long and contains a 48-bit MAC address and some control and status bits necessary to maintain the table. The table entries are updated in the following two ways:

- Automatic learning of unicast addresses performed by the switch
- CPU-initiated learning. The CPU updates the table by issuing a command in the CECR register. This mechanism is used for multicast and special unicast addresses that the CPU needs to add to the table.

If the destination address is found in the ADLT, the frame is forwarded to the port specified in the ADLT (GDA field).

For multicast traffic, the group destination address (GDA) is used for multicast grouping. The GDA field is updated by the CPU using the QUICC Engine RISC command register. See Section 40.11.1, “Command Register,” for details.

Figure 40-8 displays the ADLT entry format.

ADLT Entry Format

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	MAC Address															
OFFSET + 2	MAC Address															
OFFSET + 4	MAC Address															
OFFSET + 6	GDA0	GDA1	GDA2	GDA3	GDA4	GDA5	GDA6	GDA7	GDA8						Status	

= Reserved

[1] Offset from ADLT_BASE

Figure 40-8. ADLT Entry Format

Table 40-2 shows the address learning table bit descriptions.

Table 40-2. Address Learning Table Bit Descriptions

Field	Bits	Description
MAC Addr	0–47	MAC Address 48 bits of MAC address (Bit 40 is the multicast bit)
GDA0–8	48–56	Group Destination Address. This field is used for unicast and multicast frames. For unicast frame, this field indicates the destination port. (Only one bit is set). For multicast frame It indicates which of the ports is associated with this address. 0 Do not forward to port 0–8 1 Forward frame to port 0–8 if other criteria are met. Bit 48 corresponds to port 0, 49 to port 1...bit 56 to port 8 (CPU). This field is used in either Tag mode or Basic mode.
Reserved	57–61	Reserved. Set to 0.
Status {A,V}	62–63	Status {A,V} 00 Invalid entry (reset value) 01 Valid entry + Aging bit is zero 10 Valid entry + Locked entry 11 Valid entry + Aging bit is set

Table 40-3 maps the port number to peripheral numbers.

Table 40-3. Mapping of Port Numbers to Peripheral Numbers

Port Number	Peripheral
0	UCC1
1	UCC2
2	UCC3
3	UCC4
4	UCC5
5	UCC6
6	UCC7
7	UCC8
8	CPU

40.5.1.1 Automatic Update of ADLT Entries

The hash-based lookup provides four buckets for each entry in the ADLT table. There are two modes of operation for updating the ADLT entries.

In the first mode, the switch fills in the four buckets with the first four addresses encountered in the SA field of the incoming packets. Any additional conflicts in the buckets are ignored, and no updates occur.

In the second mode, any conflicts beyond the first four are resolved by flushing the oldest entry in the four buckets. This mode is programmed in the HCM field in Section 40.13.1, "Switch Parameters (SWPAR)."

In both of these options, the older entry might also be removed by the aging mechanism.

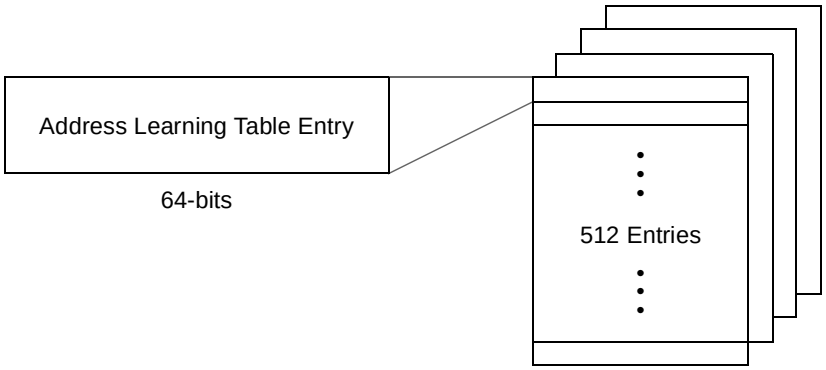


Figure 40-9. ADLT Memory

40.5.1.2 Address Aging

The purpose of the address aging mechanism is to remove unused/old entries from the ADLT and make room for new active ones. Address aging is performed by the QUICC Engine RISC (automatic aging) or by the CPU (command aging). This mechanism is provided in addition to the ADLT update mechanism.

When a new frame arrives and the switch updates the ADLT, it sets the aging bit of the relevant entry. In any aging cycle, the RISC scans 16 ADLT entries, resets all the aging bits that equal to 1 and resets the status bits of every entry containing an aging bit equal to 0 (thus, removes the entry).

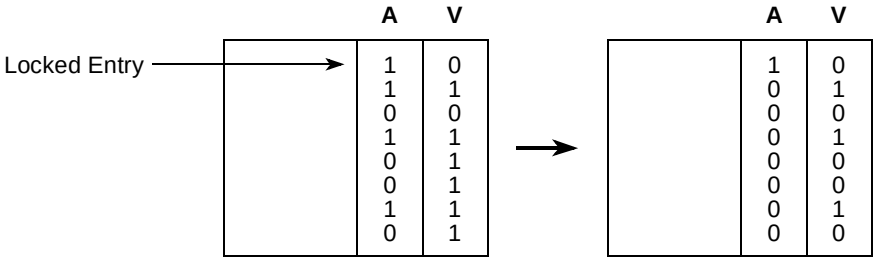


Figure 40-10. Address Aging Mechanism

The user has to determine the aging period (AP) in seconds. The ADLT entry life period, after the receipt of the last packet for this entry, is between one and two aging period seconds. For example, if the user sets the aging period to 100 seconds, the entry life period is between 100 and 200 seconds.

Before enabling the aging mechanism (See Aging Enable command, Section 40.11.3, “Switch Commands”), the CETSCR must be initialized. The CETSCR defines the time unit that is common to Time Stamps in the QUICC Engine block. Before enabling the aging, its Parameter field must be programmed.

Table 40-4. Aging Parameter Table

Offset [1]	Name	Width	Description	User Writes
0xC8	ATU	Word	Aging Time Unit. ATU = (AP * 16 * TSF)/(Total number of ADLT entries) TSF[2] = Time Stamp frequency = (QUICC Engine frequency) / (CETPS+2)	UD (User Defined)

1 Offset from Switch PRAM.
2 See CETSCR in the QUICC Engine Control chapter

Example for using AP, TSF:

AP = 100 sec

QUICC Engine frequency = 400MHz

CETPS = 998

TSF = 400MHz/1000=400KHz

Total number of ADLT entries = 2048Entries

ATU = (100*16*400,000)/2048 = 312500

40.5.1.3 Locked Entry

The CPU may lock entries in the ADLT so that the aging function does not affect them. This is accomplished by writing a command to the CECR register; refer to Section 40.11.1, “Command Register.” See Table 40-2 for the ADLT bit description.

40.5.1.4 Group Destination Address (GDA)

The Group Destination Address (GDA[0–8]) is used for unicast and multicast addresses. For unicast addresses, only one bit is set in this field and is typically updated automatically by the switch learning process. For multicast addresses, the table entry and this field are updated by the CPU. This update is application-dependent and typically occurs as a result of IGMP packet processing. IGMP packets are forwarded to the CPU by the L2 switch if the IGMP snooping feature is enabled. See Section 40.3.7, “IGMP Snooping,” for more information.

40.5.2 Port Designation Table (PDT)

The Port Designation Table (PDT) is used in tagged and untagged mode. It includes nine entries, one for each source port and one for the CPU. Each entry of the PDT contains one bit per destination port. It creates internal multicast groups for each source port.

In untagged mode it is used as a default table. Every untagged multicast frame or unicast frame whose destination address is not found in the ADLT is forwarded to the relevant ports specified by this table.

The PDT is also used in tagged mode for security problems or when the user is forced to work with it instead of the Filtering VLAN Destination Table. See Section 40.5.3.1, “Security Problems,” and Section 40.13.2, “Port Parameters (PPAR),” for more information.

NOTE

When the PDT is unused in tagged mode, the user should program the table to all ones, except the port’s own bit. See Section 40.5.5, “Forwarding Frame Back to the Source Port,” for details.

Figure 40-11 displays the entry format for the PDT.

PDT Entry Format

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0								DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8

= Reserved

1 PDT in Port/CPU PRAM.

Figure 40-11. PDT Entry Format

Table 40-5 provides a description of the PDT bits.

Table 40-5. PDT Bits Description

Field	Bits	Description
–	0–6	Reserved
Destination Port Enable	7–15	Indicates which destination port is associated with this source port (SP). 1 This port is a member of this SP group. 0 This port is not a member of this SP group. Note: Port 8 = CPU

40.5.3 VLAN Destination Table (VDT)

The VLAN Destination Table (VDT) contains programmable number of entries (typically 64). Each entry contains a 12-bit VLAN VID and one bit per source and destination port.

The VDT determines which port(s) is/are a member in each VID and is updated by the CPU using the command register. See Section 40.11.1, “Command Register,” for more information.

The source ports field (bits 12–20) is used to check if the incoming tagged frame VID is associated with this port. See Section 40.5.3.1, “Security Problems,” for more information.

Every tagged multicast frame or unicast frame whose destination address is not found in the ADLT is forwarded to the relevant ports according to the destination ports field in the VDT.

The entry format for the VDT is shown in Figure 40-12.

VDT Entry Format

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	VLAN VID												SP0	SP1	SP2	SP3
Offset	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OFFSET + 2	SP4	SP5	SP6	SP7	SP8	DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8		V

= Reserved

1 Offset from VDT_BASE

Figure 40-12. VDT Entry Format

Table 40-6 provides descriptions of the VDT bits.

Table 40-6. VDT Bits Description

Field	Bits	Description
VID	0–11	VLAN VID
Source Port En	12–20	Indicates which of the source ports are associated with this VLAN ID. 0 This source port is not a member of this VID. 1 This source port is a member of this VID. Note: Port 8 = CPU
Destination Port En	21–29	Indicates which of the destination ports are associated with this VLAN ID. 0 This destination port is not a member of this VID. 1 This destination port is a member of this VID. Note: Port 8 = CPU
–	30	Reserved
Valid	31	0 Invalid entry (reset value) 1 Valid entry

40.5.3.1 Security Problems

- If the current frame's VID is not contained in the VDT, the frame can be dropped or forwarded to the relevant ports as specified by the PDT. This is determined by the security mode in the port parameter table. See the DPT field in Section 40.13.2, "Port Parameters (PPAR)," for details.
- If, according to VDT, the current source port is not a member of its frame's VLAN, the frame is dropped.
- In Tag mode—when an untagged frame is received and NDUT bit is cleared in PPAR, the frame is dropped.

NOTE

If a security problem is found, the frame can be sent to the CPU. See the Security Debug mode field in Section 40.13.1, "Switch Parameters (SWPAR)," for details. An interrupt can then be asserted, and the CPU can update the VDT table using the command register.

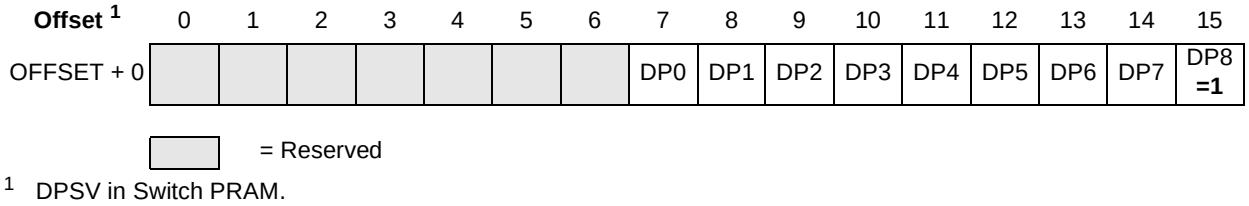
40.5.3.2 1Q Trunk

A port, designated as a 1Q trunk port, is able to receive and transmit all VIDs. In this case, there is no checking of the associated source ports in the VDT. See Section 40.13.2, “Port Parameters (PPAR),” for details.

40.5.4 Destination Port State Vector (DPSV)

The DPSV entry is updated by the CPU according to the spanning tree protocol. See Section 40.11.1, “Command Register,” for more information. It includes nine bits, one per destination port, and can prohibit the switch from forwarding packets to one or more disabled ports.

DPSV Entry Format



1 DPSV in Switch PRAM.

Figure 40-13. DPSV Entry Format

Table 40-7 provides description of the DPSV bits.

Table 40-7. DPSV Bits Description

Field	Bits	Description
–	0–6	Reserved
Destination Port En	7–15	Indicates which of the destination ports is disabled according to the spanning tree protocol. 0 This port is disabled. 1 This port is enabled.

40.5.5 Forwarding Frame Back to the Source Port

In general, the switch does not forward the frame to its source port. If the user insists on sending the frame to its source port, it can be done for each port by setting the port’s own bit in the PDT. This feature is enabled in both basic and tag modes.

40.5.6 Debug Modes

The switch supports the following debug modes:

- Error—the frame is sent to the CPU if an error is found in the incoming frame.
- Flow control—flow control packets are sent to the CPU.
- Security—the frame is sent to the CPU if a security problem is found in the incoming frame. See Section 40.5.3.1, “Security Problems,” for details.
- Miss—the frame is sent to the CPU if no destination ports are found.

These debug modes are optional modes. The user can choose to work with none, any, or all of the debug modes. See Section 40.13.1, “Switch Parameters (SWPAR),” for more information.

40.6 Egress Rule

The egress process decides if the outgoing frame is sent tagged or untagged.

In tag mode, the switch stores only tagged frames in memory. The egress process decides if the tag is removed before the frame leaves the switch.

The user can define each port as a 1D trunk. See Section 40.13.2, “Port Parameters (PPAR),” for more information. A 1D trunk is a connection from the switch that passes only untagged traffic. In this case, the switch has to remove the tag before the frame leaves that port, and the new FCS calculation replaces the old one.

40.7 Switch Processing

The L2 switch supports two processing modes—basic mode and tag mode.

All ports work in the same mode. The processing mode is set to basic mode as a default.

Basic mode is used for a single LAN environment and supports grouping based on a multicast destination address and internal port groups. See Section 40.3.7, “IGMP Snooping,” and Section 40.5.2, “Port Designation Table (PDT),” for details.

Tag mode is suitable for full VLAN and supports grouping based on a multicast destination address and VID groups. See Section 40.5.3, “VLAN Destination Table (VDT),” for more information.

In these two modes, switch processing involves the following three tasks:

- Ingress process
- Forwarding process
- Egress process

40.7.1 Basic Mode Switching

In basic mode, the L2 switch only stores standard frames (untagged frames) in memory.

The switch determines the destination ports for each received frame based on the following:

- Internal port grouping with the PDT. See Section 40.5.2, “Port Designation Table (PDT),” for details.
- Destination address as a consequence of learning or IGMP snooping packets (GDA field in the ADLT)
- Port state as a consequence of the spanning tree protocol. See the DPSV field in Section 40.5.4, “Destination Port State Vector (DPSV),” and the PS field in Section 40.13.2, “Port Parameters (PPAR),” for details

Each port is capable of passing data or management (IGMP, BPDU, and flow control packets) tagged or untagged frames.

Each port has its own ingress/egress rule based on the port parameters.

- If an error frame is found, the packet is dropped or sent only to the CPU (error debug mode), and the source address is not learned. The switch includes error counters for each port.
- When an IGMP or BPDU frame is received, the frame is sent to the CPU for processing.
- In flow control packet, the frame is dropped after being processed by the MAC of the port. The packet is sent to the CPU in Flow-Control debug mode.

40.7.1.1 Ingress Rules

Usually when a tagged frame is received, the tag is removed, and the L2 switch can consider the frame's priority. See Section 40.13.3, "Priority Mapping Table (PMT)," for more information.

However, in transparent mode (see TRA in Section 40.13.2, "Port Parameters (PPAR)") the frame is kept as is. (Note - this mode is global for all ports).

40.7.1.2 Forwarding Process

If the DA is found in the ADLT, the destination ports (DPs) are determined by GDA, PDT, and DPSV. If there are no DPs, the frame is dropped.

If the DA is not found in the ADLT, the DPs are determined by PDT and DPSV. If there are no DPs, the frame is dropped.

Figure 40-14 shows the flow chart for basic mode.

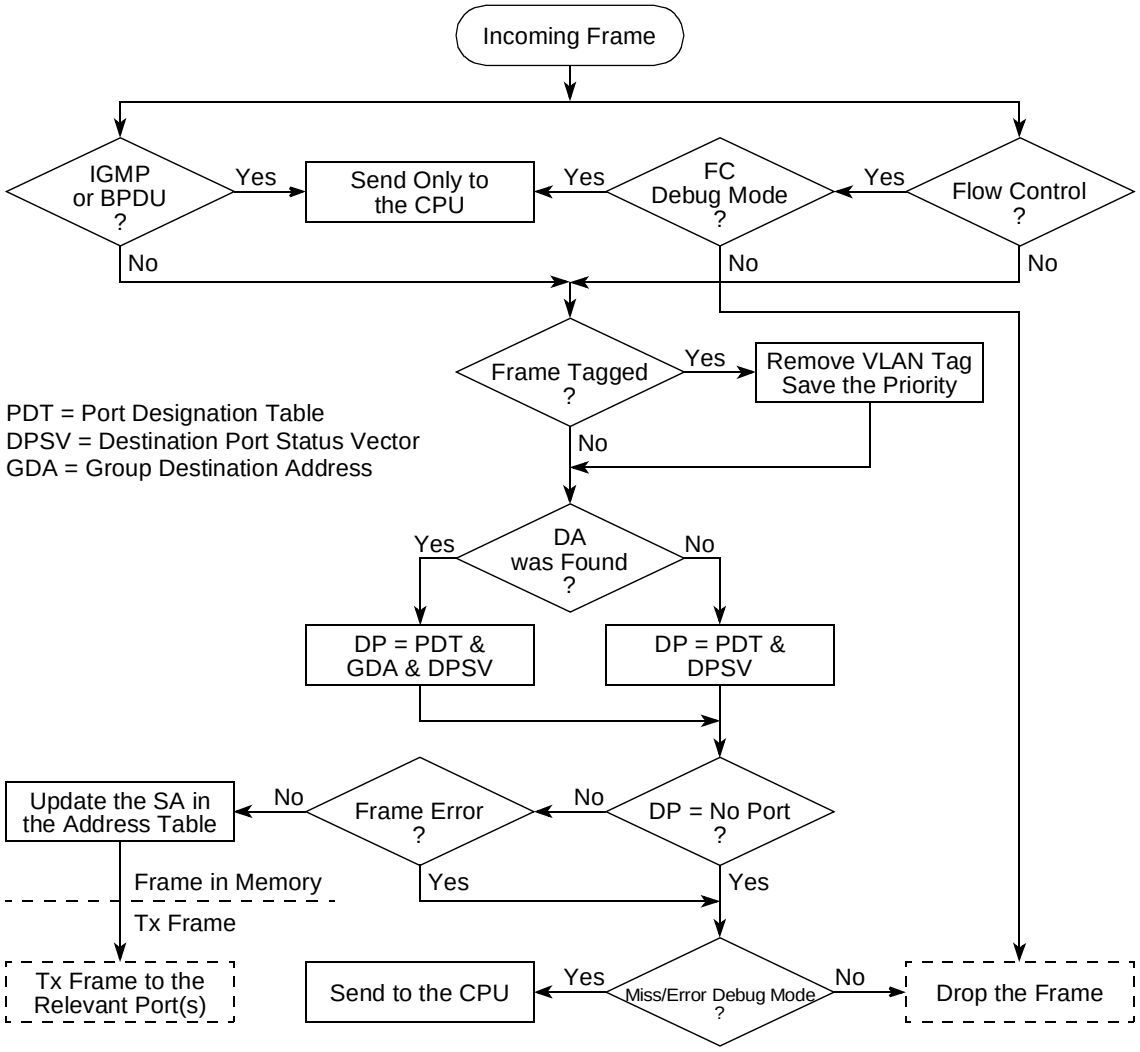


Figure 40-14. Basic Mode Flow Chart

40.7.2 Tag Mode Switching

In tag mode, the L2 switch only stores tagged frames in memory.

The switch determines the destination ports for each received frame based on the following:

- Internal ports grouping with the PDT. See Section 40.5.2, “Port Designation Table (PDT),” for details.
- VLAN grouping with the VDT. See Section 40.5.3, “VLAN Destination Table (VDT),” for details.
- Destination address as a consequence of learning or IGMP snooping packets (GDA field in the ADLT)
- Port state as a consequence of the spanning tree protocol. See the DPSV field in Section 40.5.4, “Destination Port State Vector (DPSV),” and the PS field in Section 40.13.2, “Port Parameters (PPAR),” for details.

Each port is capable of passing data or management (IGMP, BPDU, and flow control packets) tagged or untagged frames.

Each port has its own ingress/egress rule based on the port parameters.

- If an error frame is found, the packet is dropped or sent only to the CPU (error debug mode), and the source address is not learned. The switch includes error counters for each port.
- When an IGMP or BPDU frame is received, the frame is sent to the CPU for processing.
- In flow control packet, the frame is dropped after being processed by the MAC of the port. The packet is sent to the CPU in Flow-Control debug mode.

40.7.2.1 Ingress Rules

The following occurs when an untagged frame is received from any external port:

- Usually the frame is dropped or the default source port VID is added to the frame, depending on the NDUT field in Section 40.13.2, “Port Parameters (PPAR).”
- In Transparent mode (see TRA in Section 40.13.2, “Port Parameters (PPAR)”) - frame is kept as is. (Note - this mode is global for all ports).

The following occurs when a tagged frame is received:

- The tag is extracted and changed by the default VID if forced default tag mode is enabled.
- If the VID is equal to zero, it is overwritten by the default VID.
- If the VID is not contained in the VDT, the frame is either dropped or allowed to exit according the PDT.
- If the source port (SP) is not a member of the VLAN, the frame is dropped.

40.7.2.1.1 Ingress Double VLAN Tagging Rules

Double VLAN Tagging places an Extra (Service) Tag before a frame’s regular tag (additional 4 bytes).

The following occurs when RET field in Section 40.13.2, “Port Parameters (PPAR),” is set:

- The Extra Tag is removed from a received frame, and only then the port’s Ingress rules apply. Extra Tag is removed only if it exists.
- The priority and the VID used by the forwarding process are those of the inner tag.

Typically a port that has RET bit set, also has AET bit set.

40.7.2.2 Forwarding Process

The forwarding process occurs in unicast or multicast address as follows:

- If the DA is found in the ADLT, the DPs (Destination Ports) are determined according to GDA, DPSV, and PDT or VDT, depending on the DPT field in Section 40.13.2, “Port Parameters (PPAR).” If there is no DP, the frame is dropped.
- If the DA is not found in the ADLT, the DPs are determined according to DPSV and VDT or PDT. If there is no DP, the frame is dropped.

40.7.2.3 Egress Rules

When an untagged frame is received, the frame is dropped, or the default destination port VID is added to the frame (or kept as is in transparent mode).

The Egress process decides if the outgoing frames are sent tagged or untagged. The egress process refers to the 1D field in Section “Port Parameters (PPAR)”. If the value is untagged (1D trunk), the tag is removed before leaving the egress port.

40.7.2.3.1 Egress Double VLAN Tagging Rules

Typically a port that has RET bit set, also has AET bit set.

The following occurs when AET field in Section 40.13.2, “Port Parameters (PPAR),” is set:

- All frames are transmitted with Extra (Service) Tag placed after the frame‘s source address.

Figure 40-15 shows the flow chart for tag mode (Double VLAN Tagging is not shown).

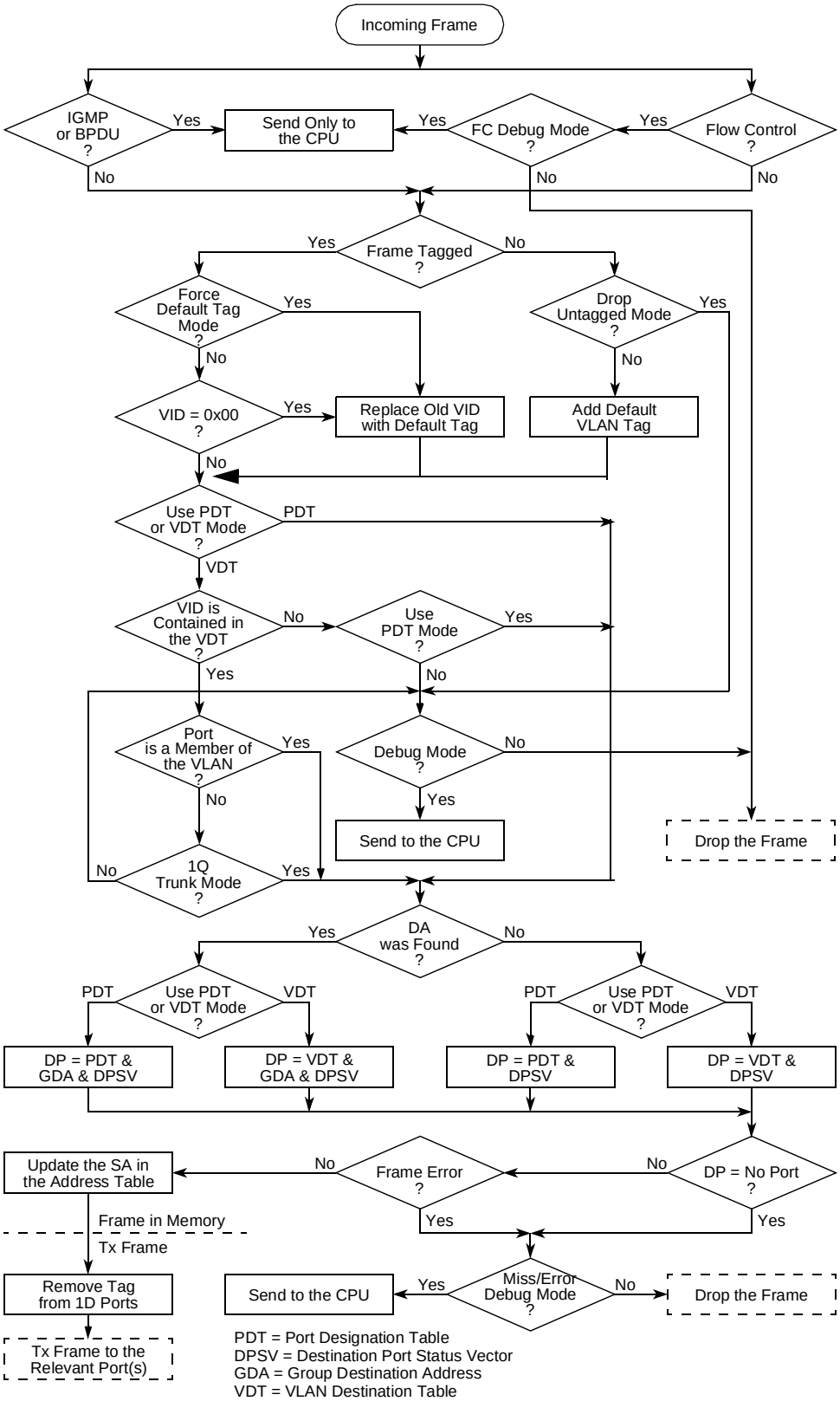


Figure 40-15. Flow Chart for TAG Mode

40.8 Switch Data Flow

The switch stores most of the packets in the external memory, up to 256 packets with a programmable maximum and minimum size.

Internal memory is used in for most of the control structure.

Each port has four priority transmit queues that limit the maximum number of packets that can be separately stored in memory for each port. The size of these transmit queues is programmable and can be smaller or even equal to the total possible packets in memory (256 packets), enabling it to support bursts of traffic.

The switch includes a frame counter for each port and one global counter for all of the ports. If one of the transmit queues is full, the next frame that arrives is dropped. If the number of frames in memory is equal to a threshold number (see Section 40.8.3, “Flow Control,” for details), the switch sends a pause packet (in full-duplex mode) or back pressure is activated (in half-duplex mode) to reduce or stop traffic. If memory is full, the next incoming frame is dropped.

40.8.1 Priority Queueing

The switch supports up to four priority queues per port or the CPU. The user can choose to use four or less priority queues for any port separately. In the case of an untagged frame without priority bits, the user can determine a default priority. See Section 40.13.2, “Port Parameters (PPAR),” for more information. Unknown unicast destination addresses are always set to lowest priority.

Every priority queue has its own size. The user must determine the size of all queues. The size of each queue varies from minimum of zero (unused queue) to maximum of 256 entries. See Section 40.8.2, “Transmit Queue Parameter Table,” for more information.

Figure 40-16 shows the transmit queue.

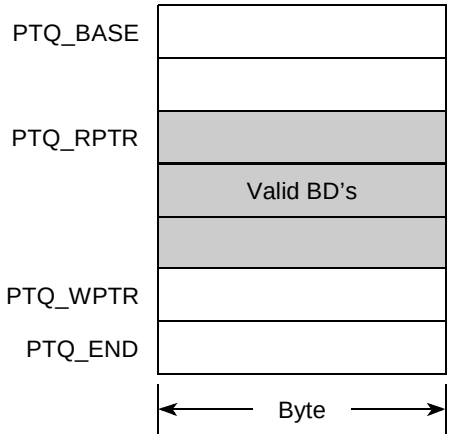


Figure 40-16. Transmit Queue

This mechanism prevents blocking between the ports. The main idea is to prevent a situation where one or several ports consume more and more buffers (packets placed in memory) as these ports may eventually become blocked entirely (all ports are blocked).

If one queue is full, the next received packet at this queue is dropped, and an interrupt is asserted.

Note that if a multicast or broadcast frame needs to be forwarded to multiple ports, and one or several of these queues are full, the frame is not dropped and is forwarded only to the queues, which are not full.

The L2 switch selects the next priority queue for transmit packets by using fixed priority, weighted round robin priority polling or a combination of them. See Section 40.8.7, "Transmission Data Structure," for more information.

40.8.2 Transmit Queue Parameter Table

Table 40-8 shows transmit queue parameters found in multi-user RAM. The TQ_BASE entry in the PRAM points to the start address of this table.

Table 40-8. Transmit Queue Parameter Table (per port)

Offset [1]	Name	Width	Description	User Writes
0x00	PTQ0_BASE	Word	Priority TQueue 0 Base pointer	User Defined Bit[31] = 0
0x04	PTQ0_END	Word	Priority TQueue 0 end pointer	Up to 256 entries
0x08	PTQ0_WPTR	Word	Priority TQueue 0 Write pointer	PTQ0_BASE
0x0C	PTQ0_RPTR	Word	Priority TQueue 0 Read pointer	PTQ0_BASE
0x10	PTQ1_BASE	Word	Priority TQueue 1 Base pointer	UD Bit[31] = 0
0x14	PTQ1_END	Word	Priority TQueue 1 End pointer.	Up to 256 entries
0x18	PTQ1_WPTR	Word	Priority TQueue 1 Write pointer	PTQ1_BASE
0x1C	PTQ1_RPTR	Word	Priority TQueue 1 Read pointer	PTQ1_BASE
0x20	PTQ2_BASE	Word	Priority TQueue 2 Base pointer	UD Bit[31] = 0
0x24	PTQ2_END	Word	Priority TQueue 2 End pointer.	Up to 256 entries
0x28	PTQ2_WPTR	Word	Priority TQueue 2 Write pointer	PTQ2_BASE
0x2C	PTQ2_RPTR	Word	Priority TQueue 2 Read pointer	PTQ2_BASE
0x30	PTQ3_BASE	Word	Priority TQueue 3 Base pointer	UD Bit[31] = 0
0x34	PTQ3_END	Word	Priority TQueue 3 End pointer.	Up to 256 entries
0x38	PTQ3_WPTR	Word	Priority TQueue 3 Write pointer	PTQ3_BASE
0x3C	PTQ3_RPTR	Word	Priority TQueue 3 Read pointer	PTQ3_BASE

[1] Offset from TQ_BASE in Port/CPU PRAM.

40.8.3 Flow Control

The purpose of flow control is to reduce the chance of dropping packets when memory is full.

Each port can operate at either half- or full-duplex mode and have flow control or back pressure enabled or disabled.

The switch uses flow control (either in full- or half-duplex) based on the port packets group limit or based on the global packets limit.

The limit value per port and the global limit for all ports are programmable. See Section 40.12.2, “Switch PRAM,” and Section 40.13.2, “Port Parameters (PPAR),” for more information on GFrame_Hlimit and PFrame_Hlimit, respectively. Therefore, there is one Rx counter per port and one global Rx counter for all receive ports. To enable burst traffic, it is recommend to set the per port packets limit to a higher number than its relative part (256/ports_number) in the memory buffers.

If the number of packets in memory that are allocated to the same source port is equal to the high port threshold number, the switch sends pause packets (in full-duplex mode) or back pressure is activated (in half-duplex mode) on this specific port to stop the traffic. When the number of packets in memory is equal to the low port threshold number, the flow control mechanism is stopped.

If the number of global frames in memory is equal to a high global threshold number, the switch sends pause packets (in full-duplex mode) or back pressure is activated (in half-duplex mode) on all the ports to reduce or stop traffic. When the number of packets in memory is equal to the low global threshold number, the flow control mechanism is stopped.

40.8.3.1 Full-Duplex Flow Control

The L2 switch supports IEEE 802.3x flow control on any full-duplex port.

When the switch sends pause packets their pause period is equal to 0xFFFF. When the number of frames in memory is equal to the low threshold, the switch sends new pause packet with pause period equal to 0x0000.

40.8.3.2 Half-Duplex Flow Control (Back Pressure)

Back pressure is used for flow control in half-duplex mode. Whenever the switch uses a flow control mechanism, the MAC of the port starts sending a JAM signal through the port. After sensing the JAM signal, the remote station defers transmission.

40.8.3.3 Receiving Pause Frame

If a port receives a pause frame with a non-zero delay, it delays transmission for the period of time specified in the received pause packet after the ongoing frame transmission is finished. If UPSMR[FTFE] bit is set and a Flow Control pause frame is received, the UCC sends the current frame till completion, and then flushes the frames in the transmit FIFO. No transmission occurs till the specified delay has elapsed.

40.8.4 Empty BD List

The empty BD list is a memory space located in the multi-user RAM. It points to the next buffer descriptor (BD) that is empty and ready for use. See Section 40.8.8, “Internal BDs,” for details.

The list is separated into two lists. The first list includes the first M BDs, and the second list includes the rest of BDs. The first M BDs point to M frames, the first 64 bytes of which are allocated in internal memory; the rest of the frames are in external memory. The last BDs point to frames that are all allocated in external memory.

The larger the value of M, the higher the performance (if internal memory is available). However, for applications where the internal memory space is not sufficient, M can be set to a small value.

For short frames (up to 128 bytes) when a new frame arrives, the switch first looks for an empty BD in the first list (high priority list). If the first list does not include any empty BD, it moves to the second one. For larger frames, it is the opposite way.

Every list has two pointers, RPTR and WPTR. The receiver follows the RPTR and reads the next empty BD. When the BD is empty again (its associated frame was transmitted), the transmitter returns the new empty BD to the list by writing after the WPTR. The valid empty BDs are located in the space between RPTR and WPTR. See Section 40.12.2, “Switch PRAM,” for more information on FEL/SEL_BASE and FEL/SEL_END.

M is equal to (FEL_END - FEL_BASE + 1).

It is required that (FEL_END - FEL_BASE + 1) + (SEL_END - SEL_BASE + 1) <= 256.

The first list is initialized with values of 0 up to (M-1) (using INIT_CPU command).

The second list is initialized with values of M up to 255 (or less) (using INIT_CPU command)

Figure 40-17 displays the two empty BD lists.

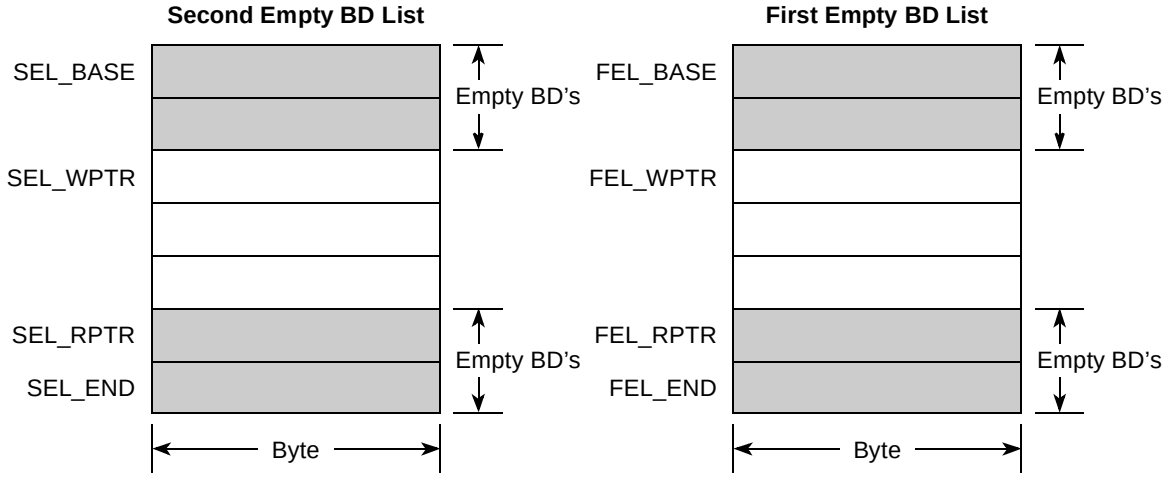


Figure 40-17. Empty BD Lists

40.8.5 BD Pointers Table

The BD pointers table includes a 32-bit frame pointer for each internal BD. See Section 40.8.8, “Internal BDs,” for more information. These pointers are used to point to the external buffer location of each frame. The table is prepared during initialization by the CPU.

The minimum spacing between two consecutive BD frame pointers has to be equal to the maximum frame length allowed. See MFLR field in Section 40.12.2, “Switch PRAM,” for more information. Also the pointers should be 8 bytes aligned.

When a new empty BD is chosen, the first 64 bytes of the new frame are stored in internal or external memory, and the rest of the frame (or all of the frame) is forwarded to external memory based on the BD pointer.

If the current frame also needs to be delivered to the CPU, all the frame is forwarded to external memory, including the first 64 bytes.

Figure 40-18 shows the BDs pointers table.

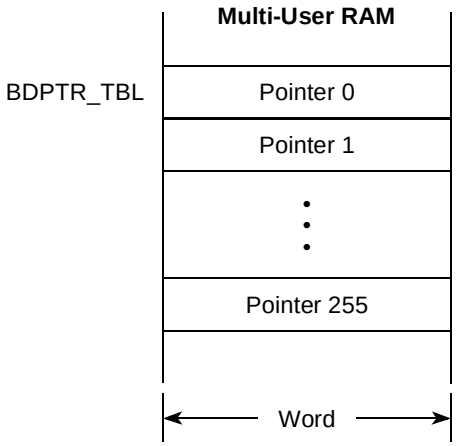


Figure 40-18. BDs Pointers Table

40.8.6 Reception Data Structure

The receiver can receive unicast, multicast or broadcast frames. When the receiver gets the first bytes of a frame, the L2 switch performs address recognition on the frame. Since Ethernet receive frame data is not written to memory until the internal address recognition algorithm completes, bus usage is not wasted on frames with empty destination ports.

If a destination port is found, the switch fetches the next empty BD and its pointer, and starts transferring the incoming frame to the BD associated data buffer. Each BD represents a full frame.

During reception, the switch checks for frames that are too short or too long. When the frame has been received, the received FCS field is checked but not written to the data buffer. The data length is written to the relevant BD in the frame length field. If an error is found (FCS error, frame is too short etc.), the BD associated with the frame is reused.

When the receive frame is completed, the BD number is written to the relevant transmit priority queue (TQ) buffer(s).

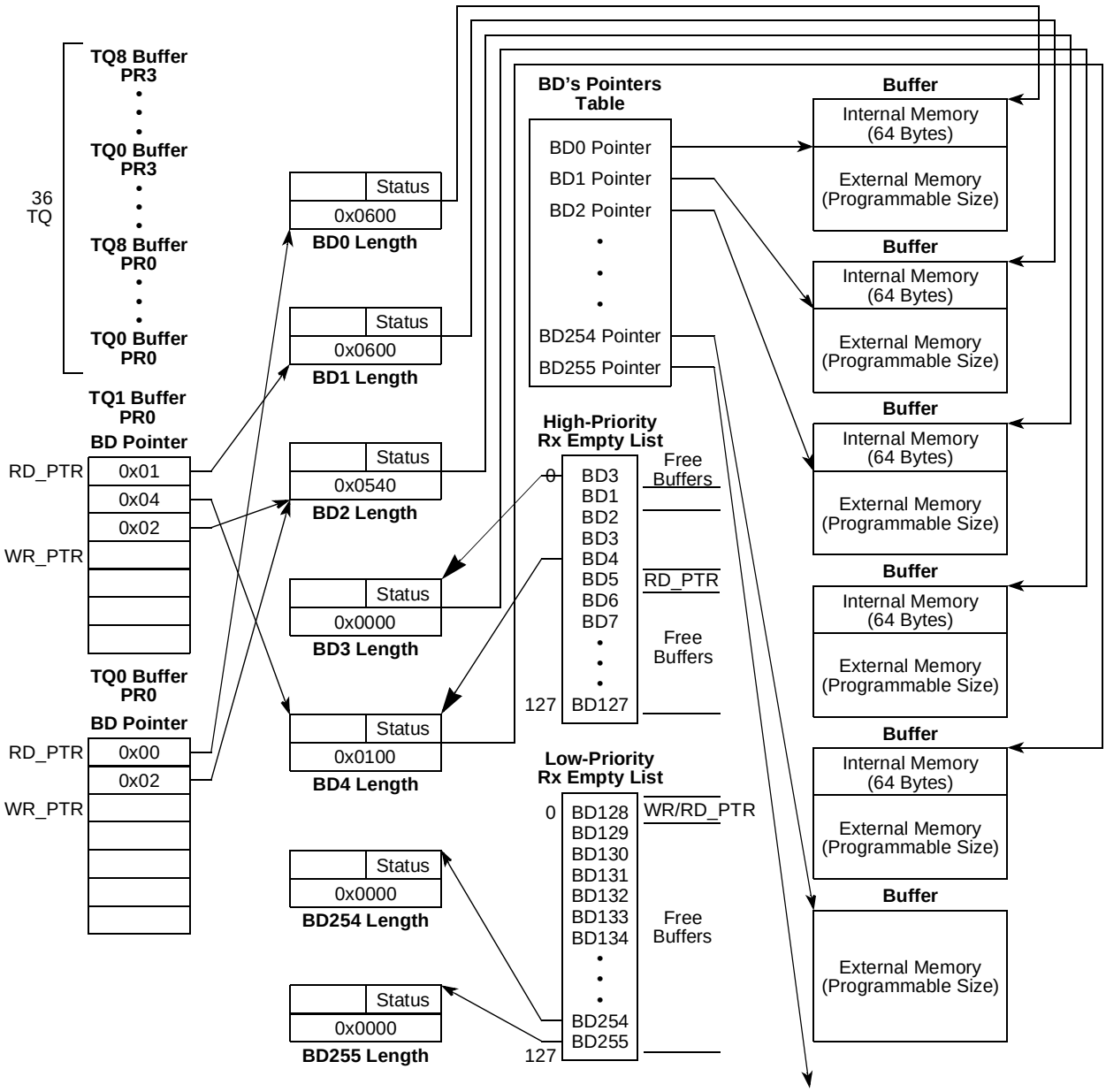


Figure 40-19. Data Structure (Without CPU Traffic)—M =128

40.8.7 Transmission Data Structure

The arbitration between TQs (with valid BDs) uses the fixed priority, weighted round robin mechanisms or a combination of them between the highest and lowest priority queues. This is determined by the PPW field in Section 40.13.2, "Port Parameters (PPAR)."

- Fixed priority mechanism—The TQ's polling always starts at TQ0. TQ0 is given the highest priority and so on down to the last TQ.

  This method can cause the lower priority TQs to starve when all frames are transmitted from the highest priority TQ.

- Weighted round robin mechanism—This mechanism gives variable priority to all TQs. All TQs have a chance to be used, preventing lower priority TQs from starving. An example for allocation may be: 16 frames from TQ0, eight from TQ1, four from TQ2, and two from TQ3.
- Fixed highest priority for TQ0 & weighted round robin for 3 lower priority queues - this allows shorter latency for TQ0.

  When a TQ is chosen, the transmitter reads the BD number from the table, and transmits a frame that is associated with this BD.

  When the end of the current buffer is reached, the transmitter releases the current BD and its pointer and writes its number back to the BD empty list.

40.8.8 Internal BDs

The L2 switch uses up to 256 BDs to report information about the received frames. These BDs are used for all external ports including the CPU and are kept in multi-user RAM.

Each BD points to a data buffer in external memory, the location of which is determined by the BD's frame pointer table. See Section 40.8.5, "BD Pointers Table," for details.

The internal BDs are located in the multi-user RAM as shown in Figure 40-20.

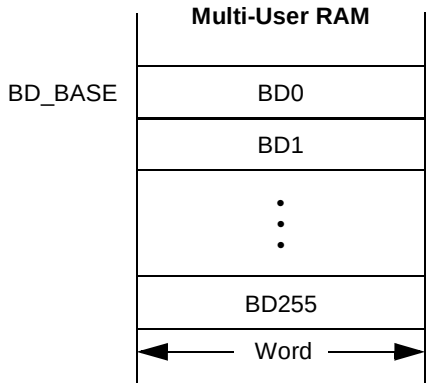


Figure 40-20. Internal BDs in Multi-User RAM

40.8.9 Packet Forwarding to the CPU

When a packet is forwarded to the CPU, the frame is stored in external memory, and the special external CPU RxBD (RCBD) is used to report information about the received frame. See Section 40.8.11.1, "Rx CPU BDs (RCBD)," for details.

If the received frame is a multicast or broadcast frame, meaning it has to be forwarded to a few external ports and to the CPU, all the frame (including the first 64 bytes) is stored in external memory.

When the whole frame is received, the switch generates a maskable interrupt, indicating that a frame was received and is in external memory. The external RCBD now points to the frame and represents it for the CPU.

A frame is forwarded to the CPU in the following scenarios:

1. The switch works in debug mode. See Section 40.5.6, “Debug Modes.”
2. It is a packet that one of the destination ports points to the CPU.
3. It is a BPDU frame.
4. It is an IGMP frame.

Frames forwarded to the CPU map to four priority levels of TQs and RCBD priority levels. See TQ_BASE and RCBD in Section 40.12.4, “CPU PRAM,” for more information.

40.8.10 Packet Forwarding from the CPU

Packets can be forwarded from the CPU to one or more external ports. The trigger for this forwarding is the CPU Tx command. See Section 40.11.1, “Command Register,” for information.

The frame length and the buffer pointer can be found in the relevant TCBD (Tx CPU BD). There are 2 types of TCBDs.

The first type is implicit (EXP=0) - the destination ports and the frame priority are determined according to the ingress rules.

The second type is explicit (EXP=1) - the destination ports and the frame priority are determined according to the bits on the TCBD itself (ingress rules are not applied).

(There is no limitation of using only one type; TCBDs may have any type).

See Section 40.8.11.2, “Tx CPU BDs (TCBD),” for information on TCBDs.

40.8.11 External CPU BDs

40.8.11.1 Rx CPU BDs (RCBD)

The L2 switch uses the Rx CPU BD (RCBD) for packet transfers to the CPU. The RCBDs are kept in external memory.

The CPU can use up to four priority RCBD tables. Each RCBD table is associated with a suitable priority transmit queue.

Each BD has 32-bit frame pointer that points to the location of the frame data buffer in external memory. The minimum spacing between two consecutive RCBD frame pointers has to be equal to the maximum frame length allowed. Every buffer stores one full frame.

Figure 40-21 shows the switch external RCBD format.

Rx CPU BUFFER DESCRIPTOR

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	E	M	W	VSP	Frame Type			SP				NO	LE	CR	OV	CL
OFFSET + 2	Frame Data Length															
OFFSET + 4	Rx Frame Buffer Pointer															
OFFSET + 6																

= Reserved

1 Offset from RCBD_PRx_BPTR in CPU PRAM - 8 bytes aligned.

Figure 40-21. External Rx CPU BD Format

Table 40-9 describes the external Rx CPU BD fields.

Table 40-9. External Rx CPU BD Format

Offset	Bits	Field	Description
0x00	0	**E**	Empty 0 The buffer associated with this RCBD is full. 1 The buffer associated with this RCBD is empty.
	1	M	Miss. Destination port was not found.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the RCBD table 1 Last BD in the RCBD table
	3	VSP	VLAN Security Problem.
	4–6	FT	Frame Type. 000 Unicast address 001 Multicast address 010 Broadcast address 011 IGMP frame. 100 Flow Control packet 101 BPDU frame. 11x Reserved
	7–10	SP	Source Port (Port 0 to Port 8) SP=0x0 - Port0 SP =0x8 - CPU
	11	NO	Rx nonoctet aligned frame
	12	LE	Rx frame length error. Long/short frame.
	13	CR	Rx FCS error. This frame contains a FCS error.
	14	OV	Overrun. A receiver overrun occurred during frame reception.
	15	CL	Collision. This frame is closed because a collision occurred during frame reception.
0x02	0–15	Data Length	Frame Data Length The data length is the number of octets the receiver writes into this BD data buffer (Not including four FCS bytes).
0x04–0x06	0–31	FBP	Rx Frame Buffer Pointer (32 bits) - 8 bytes aligned.

NOTE

At least one BD from each priority RCBD table must be prepared before beginning reception.

40.8.11.2 Tx CPU BDs (TCBD)

The switch uses the Tx CPU BD (TCBD) for packet transfers from the CPU. The TCBDs are kept in external memory.

Each BD has a 32-bit frame pointer that points to the location of the frame data buffer in external memory. Every buffer stores one full frame.

There are 2 types of TCBDs.

The first type is implicit (EXP=0) - the destination ports and the frame priority are determined according to the ingress rules.

The second type is explicit (EXP=1) - the destination ports and the frame priority are determined according to the TCBD itself (ingress rules are not applied, works like in transparent mode).

There is no limitation of using only one type; TCBDs may have any type.

NOTE

For any type, the frame data length must be in the range of MFLR and MINFLR.

Figure 40-22 shows the external Tx CPU BD format for the implicit type.

Tx CPU BUFFER DESCRIPTOR EXP=0

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	**EXP =0**	**W**			DP0	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8	PR	
OFFSET + 2	Frame Data Length															
OFFSET + 4 OFFSET + 6	Tx Frame Buffer Pointer															

= Reserved

[1] Offset from TCBD_BPTR in the CPU PRAM - 8 bytes aligned

Figure 40-22. External Tx CPU BD Format EXP=0

Table 40-10 describes the external Tx CPU BD fields for this type.

NOTE

DP[0:8] and PR should be written with 0s for the implicit type.

Table 40-10. External Tx CPU BD Format EXP=0

Offset	Bits	Field	Description
0x00	0	**E**	Empty 0 The buffer associated with this BD is full. 1 The buffer associated with this BD is empty
	1	**EXP=0**	0 - implicit type
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TCBD table 1 Last BD in the TCBD table
	5-13	DP[0:8]	(status bits when E=1) Destination Ports derived by the ingress rules (if a TQ is full then transmission will be for less ports than indicated by DP)
	14-15	PR	(status bits when E=1) Frame Priority derived by the ingress rules
0x02	0:15	**Data Length**	Frame Data Length The data length is the number of octets the transmitter should transmit from this BD data buffer (not including FCS bytes).
0x04 - 0x06	0:31	FBP	Tx Frame Buffer Pointer (32 bits) - 8 bytes aligned.

NOTE

In implicit format, if switch works in tag mode, then CPU data should include a tag (tag insertion is not done).

Figure 40-23 shows the external Tx CPU BD format for the explicit type.

Tx CPU BUFFER DESCRIPTOR EXP=1

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	**EXP =1**	**W**			**DP0**	**DP1**	**DP2**	**DP3**	**DP4**	**DP5**	**DP6**	**DP7**	0	**PR**	
OFFSET + 2	Frame Data Length															
OFFSET + 4 OFFSET + 6	Tx Frame Buffer Pointer															

= Reserved

[1] Offset from TCBD_BPTR in the CPU PRAM - 8 bytes aligned

Figure 40-23. External Tx CPU BD Format EXP=1

Table 40-11 describes the external Tx CPU BD fields for this type.

Table 40-11. External Tx CPU BD Format, EXP=1

Offset	Bits	Field	Description
0x00	0	**E**	Empty 0 The buffer associated with this BD is full. 1 The buffer associated with this BD is empty
	1	**EXP=1**	1 - explicit type
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TCBD table 1 Last BD in the TCBD table
	5-12	**DP[0:7]**	desired Destination Ports - at least one bit must be set (if a TQ is full then transmission will be for less ports than indicated by DP)

Table 40-11. External Tx CPU BD Format, EXP=1 (continued)

Offset	Bits	Field	Description
	13	**0**	Set to 0
	14-15	**PR**	desired Frame Priority
0x02	0:15	**Data Length**	Frame Data Length The data length is the number of octets the transmitter should transmit from this BD data buffer (not including FCS bytes).
0x04 – 0x06	0:31	FBP	Tx Frame Buffer Pointer (32 bits) - 8 bytes aligned.

40.9 Internal and External Memory

The switch has internal/external memories for control structure, packet buffering, the ADLT, and the VLAN table. In the maximum configuration up to 375 Kbytes (2.4 Mbytes for jumbo frames) of external memory are used. The multi-user RAM internal memory space is programmable.

Internal memory is used for Ethernet address recognition (ADLT), VLAN tables, control structure, and as for partial (high-priority) data buffering. External memory is used at most for data buffering and control structure (buffer descriptors for packets forwarded to or from the CPU).

40.9.1 Internal Memory

Parameter RAMs (PRAM)

- Up to ten PRAMs are used, 256 bytes for each PRAM
  - — Switch PRAM
  - — Ports PRAM (up to eight ports)
  - — CPU PRAM

Data Buffering

- The L2 switch supports up to 256 packet buffers. Internal memory can store the first 64 bytes of M packets.

Control Structure

- The control structure includes up to 256 local BDs, 36 transmit queues (4 TQs per port and CPU), and 2 empty BD tables.

Address Recognition

- The switch provides an on-chip ADLT with up to 2K entries for frame destination look-up.
- The switch provides an on-chip VID table and supports programmable number (typically 64) of different VIDs out of a possible 4096.

In the maximum configuration, eight external ports and 256-packet buffer, 38 Kbytes of internal multi-user RAM are used.

Table 40-12 provides the maximum size in internal memory for data buffering, control structures, address recognition, and PRAMs.

Table 40-12. Internal Memory Maximum Size (for 8 External Ports + CPU port)

Memory Area		Tables (for 256 Packets)	Size (in Bytes)
Data buffering		64 bytes * M packets	64*M
Control structures	BDs	4 bytes * 256 BDs	1024
	TQ	256 bytes * 32 TQ	8192 (worst case)
	CPU TQ	256 bytes * 4 TQ	1024 (worst case)
	Empty BD Tables	128 bytes * 2 tables	256
	Pointers Table	4 bytes * 256 BDs	1024
Address recognition	ADLT	8 bytes * 2048 entries	up to 16384
	VDT	4 bytes * 64 VIDs	256
PRAMs		256 bytes * Switch	256
		256 bytes * 8 ports	2048
		256 bytes * CPU	256
		Total	up to 30K + 64*M

40.9.1.1 Programmable Internal Memory

The overall multi-user RAM internal memory usage is programmable.

1) Table 40-13 specifies memory usage at a few configurations of ADLT size.

Table 40-13. Internal Memory Usage for ADLT

Address Table Entries	Internal Memory Usage	ADLT_MASK in Switch PRAM
128	1 Kbytes	ADLT_MASK = 1F
256	2 Kbytes	ADLT_MASK = 3F
512	4 Kbytes	ADLT_MASK = 7F
1024	8 Kbytes	ADLT_MASK = FF
2048	16 Kbytes	ADLT_MASK = 1FF

2) Table 40-14 specifies the multi-user RAM memory for one, two, three, and four queues (for at most 256 packet buffers).

Each port has four priority transmit queues; these queues limit the maximum number of packets that can be stored in memory for each port separately. The size of these transmit queues is programmable and can be smaller than or even equal to the total number of possible packets in memory (256 packets), enabling it to support bursts of traffic. So either limiting the number of priority queues or their size (256 is worst case) reduces the internal memory usage significantly.

Table 40-14. TQ Internal Memory Usage in Worst Case

Number of Priority Queues	Internal Memory Usage (per port)	Internal Memory Usage (8 Ports + CPU)
1	Up to 256 bytes	Up to 2.25 Kbytes
2	Up to 512 bytes	Up to 4.5 Kbytes
3	Up to 768 bytes	Up to 6.75 Kbytes
4	Up to 1024 bytes	Up to 9 Kbytes

3) Internal memory can store the first 64 bytes of M packets. Thus when setting the value of M, one can take into account the internal memory space limitations.

40.9.2 External Memory

Data Buffering

- The L2 switch stores most of the received packet data in external memory for up to 256 packets. The frame size is programmable.
  - — Jumbo frames are supported (up to 9 Kbytes).

Control Structure

- The control structure includes external Rx and Tx CPU BDs (RCBDs and TCBDs) for packets that are forwarded to/from the CPU.

At the maximum configuration of 256 packet buffers 375 Kbytes of external memory are used (2.4 Mbytes for jumbo frames).

40.9.3 RMON Counters

The switch can automatically gather network statistics required for Remote Monitoring (RMON).

32-bit counters per port (including the CPU), are used to provide the network management with up-to-date and historical network component data. The CPU can read, or clear these counters.

The following RMON groups are supported:

- Statistics (group 1)
- History (group 2)
- Alarm (group 3)
- Event (group 9)

The switch supports the statistics group; all other groups can be supported by the CPU.

Statistics

This statistics group contains information about the number of unicast, multicast or broadcast frames, frame rate and distribution, or the number of faulty frames classified according to error types.

The statistics group contains the following information about network load and quality:

- Packets
- Octets
- Broadcasts
- Collisions
- Dropped packets
- Fragments
- FCS alignment errors
- Undersize/oversize packets
- Multicasts
- Jabbers
- 64-octet packets
- 65- to 127-octet packets
- 128- to 255-octet packets
- 256- to 511-octet packets
- 512- to 1023-octet packets
- 1024- to 1518-octet packets

40.9.3.1 Receive RMON

The receive RMON statistics and their corresponding counters in the PRAM are described in Table 40-15.

Table 40-15. Rx RMON Counters

Offset [1]	Name	Width	Description	User Writes
0x00	RX_Byte_Cnt	Word	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).	0x0000_0000
0x04	RX_64	Word	The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).	0x0000_0000
0x08	RX_L64	Word	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) with valid FCS.	0x0000_0000
0x0C	RX_L64CRCE	Word	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) with invalid FCS.	0x0000_0000
0x10	RX_65T127	Word	The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x14	RX_128T255	Word	The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000

Table 40-15. Rx RMON Counters (continued)

Offset [1]	Name	Width	Description	User Writes
0x18	RX_256T511	Word	The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x1C	RX_512T1023	Word	The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x20	RX_1024T1522	Word	The total number of packets (including bad packets) received that were between 1024 and 1522 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x24	RX_Jabber	Word	The total number of packets received that were longer than 1522 octets (excluding framing bits but including FCS octets) and had a valid FCS	0x0000_0000
0x28	RX_JabberErr	Word	The total number of packets received that were longer than 1522 octets (excluding framing bits but including FCS octets) and had an invalid FCS	0x0000_0000
0x2C	RX_BC	Word	The total number of broadcast packets (including bad packets) received.	0x0000_0000
0x30	RX_UC	Word	The total number of unicast packets (including bad packets) received.	0x0000_0000
0x34	RX_MC	Word	The total number of multicast packets received (including bad packets).	0x0000_0000
0x38	RX_FC	Word	The total number of flow control packets received.	0x0000_0000

1 Offset from Rx_RMON_BPTR in Port PRAM / RxfromCPU_RMON_BPTR in CPU PRAM

40.9.3.2 Transmit RMON

The transmit RMON statistics and their corresponding counters in the PRAM are described in Table 40-16.

Table 40-16. Tx RMON Counters

Offset [1]	Name	Width	Description	User Writes
0x00	TX_Byte_Cnt	Word	The total number of octets of data transmitted on the network (excluding framing bits but including FCS octets).	0x0000_0000
0x04	TX_64	Word	The total number of packets transmitted that were 64 octets long (excluding framing bits but including FCS octets).	0x0000_0000
0x08	TX_L64	Word	The total number of packets transmitted that were less than 64 octets long (excluding framing bits but including FCS octets).	0x0000_0000
0x0C	TX_65T127	Word	The total number of packets transmitted that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x10	TX_128T255	Word	The total number of packets transmitted that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000

Table 40-16. Tx RMON Counters (continued)

Offset [1]	Name	Width	Description	User Writes
0x14	TX_256T511	Word	The total number of packets transmitted that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x18	TX_512T1023	Word	The total number of packets transmitted that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x1C	TX_1024T1522	Word	The total number of packets transmitted that were between 1024 and 1522 octets long inclusive (excluding framing bits but including FCS octets).	0x0000_0000
0x20	TX_Jabber	Word	The total number of packets transmitted that were longer than 1522 octets (excluding framing bits but including FCS octets).	0x0000_0000
0x24	TX_BC	Word	The total number of broadcast packets transmitted.	0x0000_0000
0x28	TX_UC	Word	The total number of unicast packets transmitted.	0x0000_0000
0x2C	TX_MC	Word	The total number of multicast packets transmitted.	0x0000_0000
0x30	TX_FC	Word	The total number of flow control packets transmitted by switch request	0x0000_0000

1 Offset from Tx_RMON_BPTR in Port PRAM / TxtoCPU_RMON_BPTR in CPU PRAM

40.9.4 Error Counters

The switch supports 32-bit error counters per port. The CPU can read, or clear these counters.

40.9.4.1 Receive Counters

The receive error statistics and their corresponding counters in the PRAM are described in Table 40-17.

Table 40-17. Rx Error Counters

Offset [1]	Name	Width	Description	User Writes
0x9C	SECEC	Word	Security Error Counter. See Section 40.5.3.1, “Security Problems” for details.	0x0000_0000
0xA0	MISSC	Word	Miss Counter. Destination port was not found.	0x0000_0000
0xA4	MFLEC	Word	MEM Full Counter. Packet was dropped due to full memory.	0x0000_0000
0xA8	CRCEC	Word	Receive FCS Error Counter	0x0000_0000
0xAC	ALEC	Word	Receive Alignment Error Counter. Rx nonoctet aligned frame arrived.	0x0000_0000
0xB0	OREC	Word	Receive Overrun Error Counter. A receiver overrun occurred during frame reception.	0x0000_0000
0xB4	COLC	Word	Receive Collision Counter. This frame was closed because a collision occurred during frame reception.	0x0000_0000

1 Offset from Port PRAM.

40.9.4.2 Transmit Counters

The transmit error statistics and their corresponding counters in the PRAM are shown in Table 40-18.

Table 40-18. Tx Error Counters

Offset [1]	Name	Width	Description	User Writes
0xDC	TxDFT_Cnt	Word	Transmit Defer Counter. Deferring while trying to transmit the frame.	0x0000_0000
0xE0	TxCRS_Cnt	Word	Transmit Carrier Sense Counter. Carrier sense lost during frame transmission.	0x0000_0000
0xE4	TxUN_Cnt	Word	Transmit Underrun Counter. Transmitter underrun condition while sending the frame.	0x0000_0000-
0xE8	TxLC_Cnt	Word	Transmit Late Collision Counter.	0x0000_0000
0xEC	TxRL_Cnt	Word	Transmit Retransmission limit Counter. The transmitter failed to successfully send a massage due to maximum allowed repeated collisions number.	0x0000_0000
0xF0	TxCL_Cnt	Word	The best estimate of the total number of collisions on this Ethernet segment.	0x0000_0000
0xF4	TxEXD_Cnt	Word	Transmit Excessive Defer Counter.	0x0000_0000

1 Offset from Port PRAM.

40.10 Switch Exceptions

40.10.1 Switch Event Register (SWE)/Mask Register (SWM)

The SWE, shown in Figure 40-24, is used as an Ethernet switch event register. This register is used for switch interaction with the CPU. It generates interrupt events recognized by the switch. On recognition of the event, the switch sets the corresponding SWE bit. Interrupts generated by this register can be masked in the Ethernet switch mask register (SWM). The proper EXT bit in CRIMR should also be set in order to generate an interrupt.

The SWM has the same bit format as the SWE. Setting a SWM bit enables and clearing a bit masks the corresponding interrupt in the SWE.

The SWE can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values. Unmasked SWE bits must be cleared before the QUICC Engine block clears the internal interrupt request.

SWITCH EVENT REGISTER / SWITCH MASK REGISTER

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	BSY	HC	CTQF	RXF0	RXF1	RXF2	RXF3	MF	TQF0	TQF1	TQF2	TQF3	TQF4	TQF5	TQF6	TQF7
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-24. Switch Event /Mask Register

Table 40-19 provides field descriptions for the switch event/mask register.

Table 40-19. Switch Event /Mask Register Field Descriptions

Bits	Field	Description
0	BSY	Busy condition. Set when a frame for CPU is received and discarded due to a lack of empty RCBDs.
1	HC	Hashing conflict occurred. (Set when frame arrives from the CPU) - when this occurs clear the whole ADLT and select new preset_crc value
2	CTQF	CPU Transmit queue(s) is full. A failure to write new frame to one (or more) CPU transmit queues occurred due to full transmit queue(s).
3	RXF0	Rx Frame Priority0. Set when a complete frame is forwarded to the CPU with priority 0.
4	RXF1	Rx Frame Priority1. Set when a complete frame is forwarded to the CPU with priority 1.
5	RXF2	Rx Frame Priority2. Set when a complete frame is forwarded to the CPU with priority 2.
6	RXF3	Rx Frame Priority3. Set when a complete frame is forwarded to the CPU with priority 3.
7	MF	Memory is full. No room for new frames. The current incoming frame was dropped. (Set when the frame has arrived from the CPU)
8	TQF0	Port0 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port0 transmit queues due to full transmit queue(s).
9	TQF1	Port1 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port1 transmit queues due to full transmit queue(s).
10	TQF2	Port2 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port2 transmit queues due to full transmit queue(s).
11	TQF3	Port3 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port3 transmit queues due to full transmit queue(s).
12	TQF4	Port4 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port4 transmit queues due to full transmit queue(s).
13	TQF5	Port5 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port5 transmit queues due to full transmit queue(s).
14	TQF6	Port6 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port6 transmit queues due to full transmit queue(s).
15	TQF7	Port7 Transmit queue(s) is full. The CPU failed to write new frame to one (or more) Port7 transmit queues due to full transmit queue(s).

40.10.1.1 SWE and SWM Address

The SWE and SWM addresses are determined by the MSNUM (SMSNUM) field in the switch parameter RAM. There are 4 possible options for the SMSNUM.

Table 40-20 maps switch MSNUM (SMSNUM) to the SWE and SWM address.

Table 40-20. Mapping of SMSNUM to SWE and SWM addresses

SMSNUM	SWE Internal Address	SWM Internal Address	External Request
0xf8	CE_BASE + 0x160 (CEEXE1)	CE_BASE + 0x164 (CEEXM1)	EXT1
0xf9	CE_BASE + 0x168 (CEEXE2)	CE_BASE + 0x16c (CEEXM2)	EXT2
0x3c	CE_BASE + 0x170 (CEEXE3)	CE_BASE + 0x174 (CEEXM3)	EXT3
0x3d	CE_BASE + 0x178 (CEEXE4)	CE_BASE + 0x17c (CEEXM4)	EXT4

NOTE

SWE/SWM refers to one of CEEXEx/CEEXMx registers that would otherwise be used by external requests. External requests registers not referred to by the SMSNUM are not used by the L2 switch.

40.10.2 Port Event Register (UCCE)/Mask Register (UCCM)

The UCCE, shown in Figure 40-25, is used as an Ethernet port event register when the UCC functions as an Ethernet switch port. It generates interrupt events recognized by the switch port. On recognition of the event, the port sets the corresponding UCCE bit. Interrupts generated by this register can be masked in the Ethernet port mask register (UCCM). The proper bit in CIMR should also be set in order to generate an interrupt.

The UCCM has the same bit format as the UCCE. Setting a UCCM bit enables and clearing a bit masks 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 the QUICC Engine block clears the internal interrupt request.

PORT EVENT REGISTER / PORT MASK REGISTER

	3	16	17	18	19	22	23	24	25	26	27	28	29	30	31
Field	CBPR	GRA	HC	CTQF	TXE	GFC	MF	TQF0	TQF1	TQF2	TQF3	TQF4	TQF5	TQF6	TQF7
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 40-25. Port Event /Mask Register

Table 40-21 provides field descriptions for the port event/mask register.

Table 40-21. Port Event /Mask Register Field Descriptions

Bits	Field	Description
3	CBPR	In half-duplex mode Change Back Pressure 0 - No change in back pressure state 1 - The UCC transmitter has entered the back pressure state; OR the UCC has exited the back pressure state. OR In full duplex mode Change Pause 0 - No change in PAUSE state. 1 - The UCC transmitter has entered the PAUSE state (i.e. it has received a PAUSE frame and its transmitter is currently not transmitting on the line); OR the UCC has exited a PAUSE state (i.e. either the MAC parameter time has expired, or a PAUSE frame with MAC parameter zero has been received). The status of the UCC transmitter is reflected in the UCCS Register.
16	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.
17	HC	Hashing conflict occurred. (Set when frame arrives from the current port) - when this occurs clear the whole ADLT and select new preset_crc value
18	CTQF	CPU Transmit queue(s) is full. The port failed to write new frame to one (or more) CPU transmit queues due to full CPU transmit queue(s).

Table 40-21. Port Event /Mask Register Field Descriptions (continued)

Bits	Field	Description
19	TXE	TX Error. A severe Tx error occurred on the transmitter channel: late collision or underrun or excessive defer or retransmission limit reached.
0-2,4-15, 20-21	-	Reserved, should be cleared.
22	GFC	Global Flow Control event. Stop CPU transmission.
23	MF	Memory is full. No room for new frames. The current incoming frame was dropped.
24	TQF0	Port0 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port0 transmit queues due to full transmit queue(s).
25	TQF1	Port1 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port1 transmit queues due to full transmit queue(s).
26	TQF2	Port2 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port2 transmit queues due to full transmit queue(s).
27	TQF3	Port3 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port3 transmit queues due to full transmit queue(s).
28	TQF4	Port4 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port4 transmit queues due to full transmit queue(s).
29	TQF5	Port5 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port5 transmit queues due to full transmit queue(s).
30	TQF6	Port6 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port6 transmit queues due to full transmit queue(s).
31	TQF7	Port7 Transmit queue(s) is full. The port failed to write new frame to one (or more) Port7 transmit queues due to full transmit queue(s).

40.10.3 Port Status Register (UCCS)

The UCCS contains the real time status of the port transmitter.

PORT STATUS REGISTER

	0	1	2	3	4	5	6	7
Field	-	-	-	-	-	-	-	BPR
Reset	0	0	0	0	0	0	0	0

Figure 40-26. Port Status Register

Table 40-22 provides field descriptions for the port status register (read only).

Table 40-22. Port Status Register Field Descriptions

Bits	Name	Description
0-6	-	Reserved
7	BPR	Back Pressure (in half-duplex mode) 0 - UCC is not in back pressure state 1 - The UCC Transmitter is in back pressure state OR Pause State (in full duplex mode) 0 - The UCC is not in PAUSE state. 1 - The UCC transmitter is in PAUSE state and is not transmitting on the line.

40.11 Switch/Ports Commands

40.11.1 Command Register

The transmit and receive commands described in the following sections are issued to the CECR.

The CPU should set CECR[FLG] when it issues a command and the QUICC Engine block clears FLG after completing the command, thus indicating to the core that it is ready for the next command. Subsequent commands to the CECR can be given only after FLG is clear.

40.11.2 Ports Commands

The QUICC Engine port command opcodes are shown in Table 40-23

Table 40-23. Ports Command Descriptions

Command	Opcode	Description
init Tx and Rx params	000000	See in CECR description (MCN=0x0d)
init Rx params	000001	See in CECR description (MCN=0x0d)
init Tx params	000010	See in CECR description (MCN=0x0d)
graceful stop Tx	000101	See in CECR description (MCN=0x0d)
restart Tx	000110	See in CECR description (MCN=0x0d)
SWITCH COMMAND	000111	Switch commands. See Section 40.11.3, “Switch Commands”

40.11.3 Switch Commands

Before issuing the command, the user should initialize the Switch command opcode (SWCO) and COMM_INFO1/2 (if needed) fields in the Switch PRAM.

For specific switch commands (SWCO=0x04/05/06/07), the core should set the SBC field (in CECR) to the desired UCC.

For general switch commands (SWCO!=0x04/05/06/07), the core can set the SBC field (in CECR) to any UCC which is a switch port.

The QUICC Engine switch command opcodes are shown in Table 40-24

Table 40-24. Switch Command Descriptions

Switch Command	Switch Command Opcode (SWCO)	COMM_ INFO1	COMM_ INFO2	Description
Update VDT Entry	0x00	VDT Entry	-	This command adds/updates one VID entry to/in the VLAN Destination Table.
VDT lookup	0x01	{VID, 20'h0}	-	This command returns the SP[0:8] and DP[0:8] of a VID in the VDT. Value is returned on COMM_INFO1[12:29] (if not found, value is 0).
Update ADLT Entry	0x02	ADLT Entry (first 32bits)	ADLT Entry (last 32bits)	This command adds/updates one entry to/in the ADLT. This command can be useful to create multicast groups as a result of IGMP snooping control packets or to create locked entries.
ADLT lookup	0x03	MAC address (first 32bits)	{last 16bit of MAC address, 0x0000}	This command returns the GDA[0:8] of a MAC address in the ADLT. Value is returned on COMM_INFO2[16:24] (if not found, value is 0).
Disabled_ Port	0x04	-	-	The port is closed for all types of frames. The ADLT is not updated.
Forwarding _Port	0x05	-	-	Regular operation (default mode)
Blocking/ Listening_ Port	0x06	-	-	The port is open (incoming and outgoing) only for BPDU frames. All other frames are dropped and the ADLT is not updated.
Learning_ Port	0x07	-	-	The port is open (incoming and outgoing) only for BPDU frames. All other frames are dropped, but the ADLT is updated for new good frames
Reset_ ADLT	0x08	-	-	This command resets the ADLT. May be used when hash conflict interrupt is asserted. During the reset period input frames are treated as frames with unknown destination addresses.
Reset_VDT	0x09	-	-	This command resets the VDT.
CPU Tx Command	0x0A	-	-	This command is used as a trigger for forwarding packet from the CPU to the switch (TCBD). The CPU uses this command when it has a full packet for the switch in external memory.
Aging Enable	0x0B	-	-	Enable the Aging mechanism. The aging routine will be done automatically by the QUICC Engine block. See Section 40.3.6, "Address Aging"
INIT_CPU	0x0C	-	-	Init parameters of CPU PRAM and some switch PRAM parameters.

40.12 Parameter RAM

40.12.1 Parameter RAM Map

The switch parameter RAM is used to configure the L2 switch. The QUICC Engine block also uses the PRAM to store operational and temporary values used during the switch activities. The switch parameter RAM is divided to 3 memory spaces. The first area (Port PRAM) serves the UCC (one space per each port), the second is dedicated for the CPU (CPU PRAM), and the last is the switch PRAM that serves all the ports and includes the basic switch operation modes.

40.12.2 Switch PRAM

The switch PRAM is mapped as shown in Table 40-25. It occupies 256 bytes. Addresses not mapped within it are reserved for QUICC Engine block internal usage.

Table 40-25. Switch PRAM Configuration

Offset	Name	Width	Description	User Writes
0x00	SWPAR	Word	Switch Parameters Register. See in Section 40.13, "Switch Tables" for more information.	UD
0x04	VDT_BASE	Word	VLAN Destination Table Base pointer	UD
0x08	VDT_MASK	Half word	This field fixes the VDT size (VDT size must be a power of 2) VDT_END = VDT_BASE + (VDT_MASK + 1)* 4 or VDT_MASK = (VDT SIZE / 4) - 1	UD
0x0A	ADLT_MASK	Half word	This field fixes the ADLT size (ADLT size must be a power of 2) ADLT_END = ADLT_BASE + (ADLT_MASK +1)* 32 or ADLT_MASK = (ADLT SIZE / 32) - 1	UD
0x0C	ADLT_BASE	Word	Address Learning Table Base pointer	UD 8 byte aligned
0x11	LOWEST_PR	Byte	Mapping of lowest priority (possible values 0:3)	UD
0x12	HIGHEST_PR	Byte	Mapping of highest priority (possible values 0:3)	UD
0x14	Internal_DBufferPTR	Word	The Base pointer for the internal data buffer (for the first M high priority BDs). This pointer points to the base buffer of BD# 0.	UD
0x18	PMT	Half word	Priority Mapping Table. See in Section 40.13, "Switch Tables" for more information.	UD
0x20	DPSV	Half word	Destination Port Status Vector. The port is open or closed for transmitting data. The value of this vector is determined by the CPU according to spanning tree (BPDU) packets. DPSV[0–6] Reserved, should be cleared. DPSV[7–15] [P0,P1,...,P7,**1**]	UD
0x22	IGDP	Half word	IGMP Destination Ports. IGMP packets are delivered to these ports: IGDP[0:6] = Reserved, should be cleared. IGDP[7:15] = [P0,P1,...,P7,CPU] Note: In case of tag mode, all these destination ports also have to be a members of the IGMP packet's VID.	UD
0x24	GFrame_Cnt	Byte	Global Frame Counter. The current number of frames in the memory.	0x00
0x25	GFrame_Hlimit	Byte	High Global Threshold. If the number of frames in memory is equal to this value, the switch starts to send pause packets to all ports.	UD
0x26	GFrame_Llimit	Byte	Low Global Threshold. If the number of frames in memory is equal to this value, the switch stops the flow control mechanism.	UD
0x27	SMSNUM	Byte	Switch MSNUM (for CPU routine)	UD
0x28	FC	Byte	FC vector [P0,P1,.....,P7]	

Table 40-25. Switch PRAM Configuration (continued)

Offset	Name	Width	Description	User Writes
0x2B	SWCO	Byte	Switch Command Opcode. See Section 40.11.3, “Switch Commands”	UD
0x2C	HConflict_cnt	Word	Hash Conflict counter	
0x34	Preset_CRC	Word	Preset value of CRC	0xFFFFFFFF
0x38	BD_BASE	Word	BDs Base pointer	UD
0x3C	BDPTR_TBL	Word	BDs Pointer Table. BDs external data pointer	UD
0x50	FEL_BASE	Word	First Empty List Base pointer (although its word aligned, bit[31] indicating full must be initialized with 1)	UD word aligned Bit[31] = 1
0x54	FEL_END	Word	First Empty List End pointer. Byte per entry.	UD
0x58	FEL_WPTR	Word	First Empty List Write pointer	{FEL_BASE [0:30]<<1}
0x5C	FEL_RPTR	Word	First Empty List Read pointer	{FEL_BASE [0:30]<<1}
0x60	SEL_BASE	Word	Second Empty List Base pointer (although its word aligned, bit[31] indicating full must be initialized with 1)	UD word aligned Bit[31] = 1
0x64	SEL_END	Word	Second Empty List End pointer. Byte per entry	UD
0x68	SEL_WPTR	Word	Second Empty List Write pointer	{SEL_BASE [0:30]<<1}
0x6C	SEL_RPTR	Word	Second Empty List Read pointer	{SEL_BASE [0:30]<<1}
0x70	IPMT	Word	IP Priority Mapping Table. PR[0:15] See Section 40.13, “Switch Tables” for more information.	UD
0x74	IPMT	Word	IP Priority Mapping Table. PR[16:31]	UD
0x78	IPMT	Word	IP Priority Mapping Table. PR[32:47]	UD
0x7C	IPMT	Word	IP Priority Mapping Table. PR[48:63]	UD
0xBC	MFLR	Half word	Maximum frame length allowed (not including four FCS bytes) Typically 1514 in basic mode or 1518 in tag mode. Jumbo frames are also supported.	UD
0xBE	MINFLR	Half word	Minimum Frame Length allowed, not including FCS (typically 60 decimal).	UD
0xC4	Next_Entry	word	The next aging entry	ADLT_BASE
0xC8	ATU	word	Aging Time Unit. See Section 40.3.6, “Address Aging”	UD
0xD0	COMM_INFO1	Word	This field is used for command information. First 32 bits.See Section 40.11.3, “Switch Commands”	UD
0xD4	COMM_INFO2	Word	This field is used for command information. Second 32 bits. See Section 40.11.3, “Switch Commands”	UD
0xDC	EXRAM_PORT0	Word	Port0 PRAM base pointer.	0x0000_8400

Table 40-25. Switch PRAM Configuration (continued)

Offset	Name	Width	Description	User Writes
0xE0	EXRAM_PORT1	Word	Port1 PRAM base pointer.	0x0000_8500
0xE4	EXRAM_PORT2	Word	Port2 PRAM base pointer.	0x0000_8600
0xE8	EXRAM_PORT3	Word	Port3 PRAM base pointer.	0x0000_9000
0xEC	EXRAM_PORT4	Word	Port4 PRAM base pointer.	0x0000_8000
0xF0	EXRAM_PORT5	Word	Port5 PRAM base pointer.	0x0000_8100
0xF4	EXRAM_PORT6	Word	Port6 PRAM base pointer.	0x0000_8200
0xF8	EXRAM_PORT7	Word	Port7 PRAM base pointer.	0x0000_8300
0xFC	EXRAM_CPU	Word	Port8 (CPU) PRAM base pointer	UD

40.12.3 Port (UCC) PRAM

Each UCC PRAM area begins at the same offset from each UCC base area.

The UCC PRAM is mapped as shown in Table 40-26.

Table 40-26. Port PRAM Configuration

Offset	Name	Width	Description	User Writes
0x00	PPAR	Word	Port Parameters Table. See Section 40.13, “Switch Tables” for more information.	UD
0x04	RISC ALLOCATION	Byte	Risc allocated for the UCC 0x01 - RISC1 0x02 - RISC2	UD
0x05	RXIDLE	Byte	Internal variable	
0x08	RSTATE	Word	bus mode and Rx internal state	BUSMode,24’h0
0x0C	TSTATE	Word	bus mode and Tx internal state	BUSMode,24’h0
0x18	SWPRAM	Word	Switch PRAM Base pointer	UD
0x1C	DTTE	Byte	Internal variable. Initialize to value = 0x10	0x10
0x1D	DMFF	Byte	Internal variable. Initialize to value = 0xFF	0xFF
0x1E	DTTL	Byte	Internal variable. Initialize to value = 0x40	0x40
0x34	Port_Mirror	Half Word	a bit set for one of the ports makes it a mirror of the source port. [7’b0,P0,P1,.....,P7,CPU]	UD
0x80	PDT	Half word	Port Designation Table PDT[0–6] Reserved, should be cleared. PDT[7–15] [P0,P1,...,P7,CPU]	UD PDT[0–6]=0
0x90	Type_or_Len	Half Word	Internal variable. Initialize to value = 0x0600	0x0600
0x92	DTag	Half word	Default Tag = [PR(3bits),CFI(1bit),VID(12bits)] In tag mode, the switch may add or replace the original tag with the default tag. Each port has its own default tag (VID and priority).	UD
0x94	PFrame_Cnt	Byte	Port Frame Counter.The number of the frames (that arrived from this port) in the memory.	0x00

Table 40-26. Port PRAM Configuration (continued)

Offset	Name	Width	Description	User Writes
0x95	PFrame_Hlimit	Byte	High Port Threshold. If the number of frames from this source port in memory is equal to this value, the switch starts to send pause packets to this SP.	UD
0x96	PFrame_Llimit	Byte	Low Port Threshold. The switch stops the Flow Control mechanism when the number of frames (from this SP) in memory is equal to this value.	UD
0x98	Extra Tag TYPE	Half Word	Extra (Service) Tag Type (distinct from 0x8100)	UD
0x9a	Extra Tag ID	Half word	Extra (Service) Tag ID	UD
0x9C	SECEC	Word	Security Error Counter. See Section 40.5.3.1, “Security Problems”	
0xA0	MISSC	Word	Miss Counter. Destination port was not found.	
0xA4	MFLEC	Word	MEM Full Counter. Packet dropped due to full memory.	
0xA8	CRCEC	Word	Receive FCS Error Counter.	
0xAC	ALEC	Word	Receive Alignment Error Counter.	
0xB0	OREC	Word	Receive Overrun Error Counter.	
0xB4	COLC	Word	Receive Collision Counter (a collision occurred during frame reception)	
0xC0	TQ_BASE	Word	TQ Table Base pointer	UD 8 byte aligned
0xC4	TQ0_MAX_CNT	Byte	TQ0 Weighted Priority. Maximum continuous frames from TQ0.	UD
0xC5	TQ1_MAX_CNT	Byte	TQ1 Weighted Priority. Maximum continuous frames from TQ1.	UD
0xC6	TQ2_MAX_CNT	Byte	TQ2 Weighted Priority. Maximum continuous frames from TQ2.	UD
0xC7	TQ3_MAX_CNT	Byte	TQ3 Weighted Priority. Maximum continuous frames from TQ3.	UD
0xDC	TxDFT_Cnt	Word	Transmit Defer Counter. Deferring while trying to transmit the frame.	
0xE0	TxCRS_Cnt	Word	Transmit Carrier Sense Counter. Carrier sense lost during frame transmission.	
0xE4	TxUN_Cnt	Word	Transmit Under Run Counter. Transmitter underrun condition while sending the frame.	
0xE8	TxLC_Cnt	Word	Transmit Late Collision Counter.	
0xEC	TxRL_Cnt	Word	Transmit Retransmission limit Counter. The transmitter failed to successfully send a massage due to maximum allowed repeated collisions number.	
0xF0	TxCL_Cnt	Word	The best estimate of the total number of collisions on this Ethernet segment.	
0xF4	TxEXD_Cnt	Word	Transmit Excessive Defer Counter.	
0xF8	Rx_RMON_BPTR	Word	Receive RMON Base pointer.	UD
0xFC	Tx_RMON_BPTR	Word	Transmit RMON Base pointer	UD

40.12.4 CPU PRAM

The CPU PRAM is mapped as shown in Table 40-27

Table 40-27. CPU PRAM Configuration

Offset	Name	Width	Description	User Writes
0x00	PPAR	Word	CPU Parameters Table. See Section 40.13, "Switch Tables" for more information.	UD
0x08	RSTATE	Word	bus mode and Rx internal state	BUSMode,24'h0
0x0C	TSTATE	Word	bus mode and Tx internal state	BUSMode,24'h0
0x30	Rx_TEMP_PTR	Word	Receive TEMP internal pointer (64 bytes buffer) for use with TCBDs	UD
0x34	Port_Mirror	Half Word	a bit set for one of the ports makes it a mirror of the source port. [7'b0,P0,P1,.....,P7,1'b0]	UD
0x40	RCBD_PR0_BPTR	Word	RCBD Priority0 Base Pointer	UD
0x48	RCBD_PR1_BPTR	Word	RCBD Priority1 Base Pointer	UD
0x50	RCBD_PR2_BPTR	Word	RCBD Priority2 Base Pointer	UD
0x58	RCBD_PR3_BPTR	Word	RCBD Priority3 Base Pointer	UD
0x80	PDT	Half word	Port Designation Table PDT[0–6] = PDT[15] = 0 PDT[7–14] = [P0,P1,...,P7]	UD PDT[0:6] = 0 PDT[15] = 0
0x90	Type_or_Len	Half Word	Internal variable. Initialize to value = 0x0600	0x0600
0x92	DTag	Half word	Default Tag = [PR(3bits),CFI(1bit),VID(12bits)] In tag mode, the switch may add or replace the original tag with the default tag. Each port has its own default tag (VID and priority).	UD
0x94	PFrame_Cnt	Byte	Port (CPU) Frame Counter. The number of the frames (that arrived from this port) in the memory.	0x00
0x98	Extra Tag TYPE	Half Word	Extra (Service) Tag Type (distinct from 0x8100)	UD
0x9a	Extra Tag ID	Half word	Extra (Service) Tag ID	UD
0x9C	SECEC	Word	Security Error Counter. See Section 40.5.3.1, "Security Problems."	
0xA0	MISSC	Word	Miss Counter. Destination port was not found.	
0xA4	MFLEC	Word	MEM Full Counter. Packet was dropped due to full memory.	
0xB0	TCBD_BPTR	Word	TCBD Base Pointer in external memory.	UD
0xB4	TCBD_CPTR	Word	TCBD Current Pointer	TCBD_BPTR
0xC0	TQ_BASE	Word	CPU TQ Table Base pointer.	UD 8 byte aligned
0xC4	TQ0_MAX_CNT	Byte	CPU TQ0 Weighted Priority. Maximum continuous frames from TQ0.	UD
0xC5	TQ1_MAX_CNT	Byte	CPU TQ1 Weighted Priority. Maximum continuous frames from TQ1.	UD
0xC6	TQ2_MAX_CNT	Byte	CPU TQ2 Weighted Priority. Maximum continuous frames from TQ2.	UD
0xC7	TQ3_MAX_CNT	Byte	CPU TQ3 Weighted Priority. Maximum continuous frames from TQ3.	UD
0xF8	RxfromCPU_ RMON_BPTR	Word	Receive from CPU RMON Base pointer	UD
0xFC	TxtoCPU_ RMON_BPTR	Word	Transmit to CPU RMON Base pointer	UD

40.13 Switch Tables

40.13.1 Switch Parameters (SWPAR)

SWITCH PARAMETERS

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	SM		EDM	FDM	SDM	MDM							HCM	RMON	AG	
0x02																

= Reserved

1 SWPAR in switch PRAM.

Figure 40-27. Switch Parameters

Table 40-28. Switch Parameters Table Entry

Bits	Field	Description
0	SM	Switch Mode 0 Basic mode (default mode) 1 Tag mode
2	EDM	Error Debug Mode. If an error is found in the incoming frame, it is sent to the CPU (frame should have no security problem)
3	FDM	Flow Control Debug Mode. Flow control packets are sent to the CPU.
4	SDM	Security Debug mode. If a security problem is found in the incoming frame, it is sent to the CPU.
5	MDM	Miss Debug Mode. If no destination port is found, the frame is sent to the CPU. (frame should have no security problem)
12	HCM	Hash Conflict Mode. When a hashing conflict happens: 0 The switch keeps the old four active entries in ADLT and floods the new ones. 1 The address that has not been seen for the longest period of time is discarded and replaced by the new address in ADLT.
13	RMON	RMON 0 RMON is disabled. 1 RMON is enabled.
14	AG (Read Only bit)	Aging Enable. 0 The Aging mechanism is disabled. 1 The Aging mechanism is enabled Note: For enabling this mode, the user must use Aging enable switch command. See Section 40.11.3, “Switch Commands.”

40.13.2 Port Parameters (PPAR)

PORT PARAMETERS

Offset [1]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	DPT		1Q	1D	PS		NDUT	FDT	IGE	STE	AET	BCD	IPP		PPW	
0x02	RET	TRA														

= Reserved

[1] PPAR in Port/CPU PRAM.

Figure 40-28. Port Parameters

Table 40-29. Port Parameters Table Entry

Bits	Field	Description
0–1	DPT	Destination Port Table selection. (Security Mode) 00 Use VDT (Only in tag mode). Discard the incoming frame if its VID not contained in the VDT. 01 Use VDT (Only in tag mode); Use PDT for the incoming frame if its VID is not contained in the VDT. 10 Use PDT (basic or tag mode) 11 Reserved
2	1Q	1Q Trunk. A port designated as a 1Q trunk port is enable to receive and transmit all VIDs (no checking of the associated source ports in VDT) - (only in Tag mode). 0 Normal port 1 1Q trunk
3	1D	1D Trunk. 1D trunk is a connection from the switch that passes only untagged traffic - (only in Tag mode). 0 Normal port 1 1D trunk Note: If the CPU is the source port, this field should be cleared.
4–5	PS	Port State. Each port can work in a different state as a consequence of the spanning tree protocol. 00 Disabled —The port is closed for any types of frames. The ADLT and the RMON counters are not updated. 01 Forwarding —Regular operation (default mode) 10 Blocking/Listening —The port is open (incoming and outgoing) only for BPDU frames. All other frames are dropped and the ADLT and the RMON counters are not updated. 11 Learning—The port is open (incoming and outgoing) only for BPDU frames. All other frames are dropped, but the ADLT is updated for any new good frame. Note: It is recommend to activate these modes by using switch command. See Section 40.11.3, “Switch Commands” Note: for the CPU source port, this field is set to 01 (Forwarding) by the firmware.
6	NDUT	Drop Untagged Frames - (only in Tag mode) 0 Drop untagged frames 1 Add default tag (VID + priority) to untagged frames
7	FDT	Force Default Tag (VID) - (only in Tag mode) 0 Do not force default tag 1 Force default tag
8	IGE	IGMP Enable. The switch can support IGMP snooping per port. 0 IGMP disable. IGMP packet is parsed as a normal frame. 1 IGMP enable
9	STE	Spanning Tree Enable. The switch can support spanning tree packets per port. 0 Spanning tree disable. BPDU packet is parsed as a normal frame. 1 Spanning tree enable

Table 40-29. Port Parameters Table Entry (continued)

Bits	Field	Description
10	AET	Add Extra (Service) Tag - (only in Tag mode) 1 - all frames are transmitted with Extra tag - used for double-tagging.
11	BCD	Broadcast Disable. The switch can drop all Broadcast frames in case of Broadcast storm. 0 Normal operation. 1 The port is closed for Broadcast traffic. The ADLT is updated for any new good frame. Note: If the CPU is the source port, this field should be 0.
12–13	IPP	IP Packets Priority 00 Use IP fields for priority mapping (for tagged and untagged frames) 01 Use IP fields for priority mapping only in case of untagged frame and tag priority in case of tagged frames. 10 Ignore IP priority (IPv4 & IPv6 priority fields). 11 Force default priority.
14-15	PPW[0:1]	Port Priority Weight. The switch uses fixed priority and/or Weighted Round Robin mechanism between the highest and the lowest priority queues: 00 Fixed priority. Polling always starts at TQ# 0. 01 Reserved. 10 Use a X0, X1, X2, X3 weighted priority. See the TQx_MAX_CNT field in Section 40.12.3, “Port (UCC) PRAM and Section 40.12.4, “CPU PRAM” for details. X0 = Max_TQ0_cnt X1 = Max_TQ1_cnt X2 = Max_TQ2_cnt X3 = Max_TQ3_cnt 11 Fixed priority and WRR - Polling always starts at TQ# 0, WRR for other queues. Use a X1, X2, X3 weighted priority. X1 = Max_TQ1_cnt X2 = Max_TQ2_cnt X3 = Max_TQ3_cnt
16	RET	Remove Extra (Service) Tag - (only in Tag mode) - used for double-tagging 1 - when a frame is received with 2 tags, the first tag is ommitted. When a frame is received with one tag, it is ommitted.
17	TRA	0 - no transparent mode 1 - Transparent mode - frame kept as is (no addition/removal of tag) (if TRA=1, bits RET,AET,FDT,1D should be cleared)
18-31	Reserved	Must be cleared

40.13.3 Priority Mapping Table (PMT)

PRIORITY MAPPING TABLE

Offset [1]	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
0x00	Tag_0x0	Tag_0x1	Tag_0x2	Tag_0x3	Tag_0x4	Tag_0x5	Tag_0x6	Tag_0x7

= Reserved

1 PMT in switch PRAM.

Figure 40-29. Priority Mapping Table

Table 40-30. Priority Mapping Table Entry

Bits	Field	Description
0–1	Tag_0x0	If the original packet’s priority is equal to 0x0, it is mapped to this field value.
2–3	Tag_0x1	If the original packet’s priority is equal to 0x1, it is mapped to this field value.

Table 40-30. Priority Mapping Table Entry (continued)

Bits	Field	Description
4–5	Tag_0x2	If the original packet's priority is equal to 0x2, it is mapped to this field value.
6–7	Tag_0x3	If the original packet's priority is equal to 0x3, it is mapped to this field value.
8–9	Tag_0x4	If the original packet's priority is equal to 0x4, it is mapped to this field value.
10–11	Tag_0x5	If the original packet's priority is equal to 0x5, it is mapped to this field value.
12–13	Tag_0x6	If the original packet's priority is equal to 0x6, it is mapped to this field value.
14–15	Tag_0x7	If the original packet's priority is equal to 0x7, it is mapped to this field value.

40.13.4 IP Priority Mapping Table (IPMT)

IP PRIORITY MAPPING TABLE

Offset [1]	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
0x00	PR_0x0	PR_0x1	PR_0x2	PR_0x3	PR_0x4	PR_0x5	PR_0x6	PR_0x7
0x02	PR_0x8	PR_0x9	PR_0x10	PR_0x11	PR_0x12	PR_0x13	PR_0x14	PR_0x15
0x04	PR_0x16	PR_0x17	PR_0x18	PR_0x19	PR_0x20	PR_0x21	PR_0x22	PR_0x23
0x06	PR_0x24	PR_0x25	PR_0x26	PR_0x27	PR_0x28	PR_0x29	PR_0x30	PR_0x31
0x08	PR_0x32	PR_0x33	PR_0x34	PR_0x35	PR_0x36	PR_0x37	PR_0x38	PR_0x39
0x0a	PR_0x40	PR_0x41	PR_0x42	PR_0x43	PR_0x44	PR_0x45	PR_0x46	PR_0x47
0x0c	PR_0x48	PR_0x49	PR_0x50	PR_0x51	PR_0x52	PR_0x53	PR_0x54	PR_0x55
0x0e	PR_0x56	PR_0x57	PR_0x58	PR_0x59	PR_0x60	PR_0x61	PR_0x62	PR_0x63

= Reserved

1 IPMT in switch PRAM.

Figure 40-30. IP Priority Mapping Table

Table 40-31. IP Priority Mapping Table Entry

Offset	Bits	Field	Description
0x00	0–1	IP_0x0	If the original IP packet's priority is equal to 0x0, it is mapped to this field value.
0x00	2–3	IP_0x1	If the original IP packet's priority is equal to 0x1, it is mapped to this field value.
. . . .	. . . .	. . . .	. . . .
0x10	14–15	IP_0x63	If the original IP packet's priority is equal to 0x63, it is mapped to this field value.

40.13.5 Bus Mode Register

This register was named ‘FCR’ in MPC82xx.

Figure 40-31 shows the format of the bus mode registers, which reside at TSTATE[0–7] and RSTATE[0–7] in Port/CPU PRAM.

Bits	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	DTB	BDB

Figure 40-31. Bus Mode Register (BMR)

BMR fields are described in Table 40-32.

Table 40-32. BMR 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. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 24.1.8.2, “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.
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.
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	BDB	Indicates on what bus the BDs are located. 0 On the coherent system bus (CSB). 1 On the QUICC Engine secondary bus.

40.14 L2 Switch Initialization

The following fields should be programmed as described below when using the L2 switch:

1. PT in UEMPR should have a realistic value - its recommended not using the max value.
2. SCOV=0x01 in UESCR.
3. URMODE = UTMODE = 1 in GUEMR - fast protocol.
4. MODE = 0xC in GUMR.
5. CRE=1, PAD/CRC=0 in MACCFG2 register.
6. TERCE = 1, FTFE = 1 in UPSMR.

When programming the switch it is advised to do the following steps:

1) configure the I/O ports

2) configure the UCCs registers except for ENR,ENT bits in GUMR

3) initialize all the user-defined fields in the switch, ports and CPU PRAMs

4) execute init Tx and Rx commands for every switch port

5) execute INIT_CPU command

6) enable the UCCs by setting ENR,ENT bits in GUMR.

40.15 L2 Switch Port Removal

When removing a port from the L2 switch, it is advised to do the following steps:

1) execute Disabled_Port switch host command

2) check that RXIDLE in Port PRAM is equal to 0x00

3) check that all 4 TQs of the port are empty; i.e. check that

((PTQxWPTR==PTQxRPTR)&&~PTQx_BASE[31]) == 1 for x=0:3.

4) Reset ENR,ENT in UCC GUMR.

Chapter 41
E2AAL2 Microcode

This implementation of the ATM adaptation layer type 2 (AAL2) is compliant with the ITU-T recommendations I.363.2 and I.366.1. This section describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR and is meant to be used as a supplement to the ATM controller chapter, see Chapter 32, "ATM Controller AAL0, AAL1, and AAL5."

41.1 Introduction

AAL2 enables the multiplexing of voice and data channels over a single ATM VC. The channels consist of packets transported within individual ATM cells (see Figure 41-1). Packet lengths are allowed to vary in order to accommodate bandwidth fluctuations of the individual channels. Each packet has a channel identifier (CID) so that each AAL2 user (channel) is uniquely identified by the triplet VP | VC | CID.

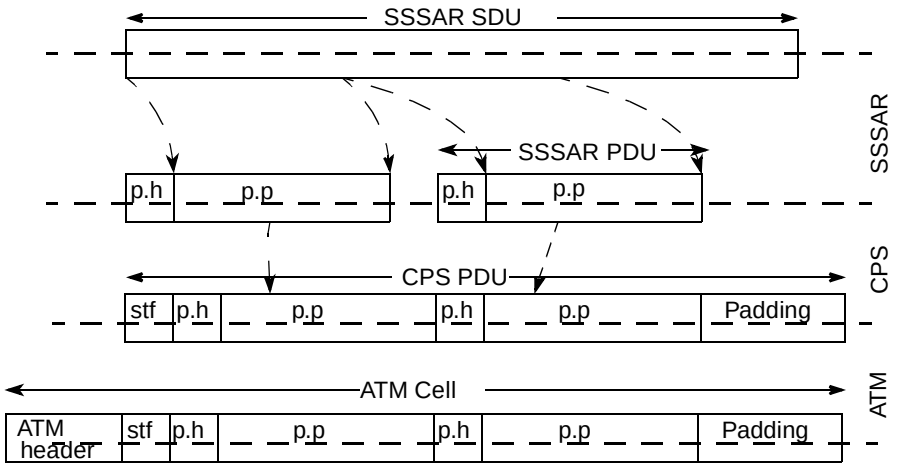


Figure 41-1. AAL2 Data Units

AAL2 is subdivided into two sublayers (see Figure 41-2): the common part sublayer (CPS) and the service-specific convergence sublayer (SSCS). In the CPS sublayer, variable length packets coming from multiple users are assembled into CPS-PDUs belonging to a single ATM VCC. The SSCS sublayer handles the mapping of user data to the CPS sublayer. The SSCS segments large data frames into smaller CPS packets and also provides different services to the user, such as transmission error detection. The SSCS sublayer is further divided into three service-specific layers: the service-specific segmentation and reassembly sublayer (SSSAR), the service-specific transmission error detection sublayer (SSTED), and the service-specific assured data transfer sublayer (SSADT).

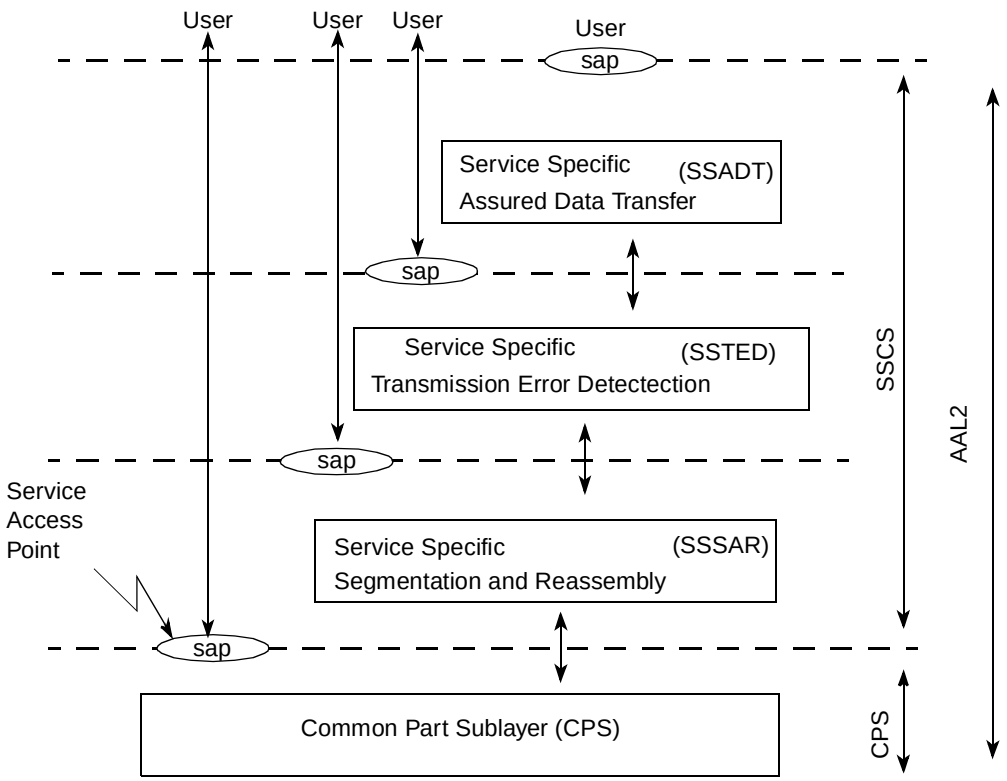


Figure 41-2. AAL2 Sublayer Structure

The AAL2 microcode implements the CPS and SSSAR sublayers. (The SSADT and SSTED sublayers are not implemented.) As shown in Figure 41-2, the user can access the CPS sublayer directly or through the SSSAR sublayer. The SSSAR sublayer is used mainly for transferring large data frames.

The AAL2 microcode also enables switching from one PHY | VP | VC | CID combination to another; an example is shown in Figure 41-3.

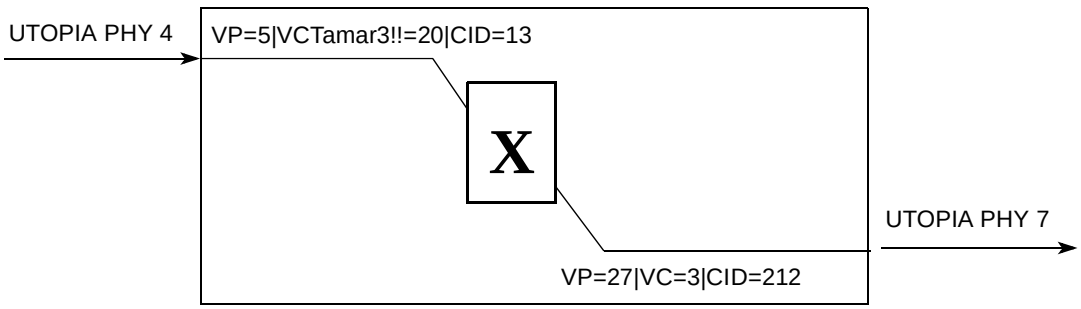


Figure 41-3. AAL2 Switching Example

41.2 Features

- Fully complies with ITU-T I.363.2 and ITU-T I.366.1 specifications.
- Supports up to 256 internal channels of AAL2
- Supports CBR, VBR, UBR+ and GBR traffic types.
  - PCR pacing
  - VBR pacing
  - UBR+ pacing
  - GBR pacing
- Priority mechanism for transmitting per VC. This allows for flexibility in prioritization of the Tx Queues.
- CNT_CU support.
- NoSTF mode support.
- Support for partially filled cells.
- User-defined cells (as described in the UDC section of the ATM controller chapter, see Section 32.2.16, "User-Defined Cells (UDC)."
- Interrupt indications include the ATM channel number, the CID, and the event type. The events reported are TX buffer not ready, TX buffer transmitted, RX buffer not ready, RX buffer, RX SSSAR frame, and RX AAL2 error events.
- Support CID MASKING - allows CID screening on a per channel basis. The packets with CID outside the mask are discarded.
- CPS switching
  - Switching from a receive PHY_1 | VP_1 | VC_1 | CID_1 combination to another transmit PHY_2 | VP_2 | VC_2 | CID_2 combination.
  - Receive one or multiple VP | VC | CIDs directly to a specific TX queue to enable switching.
  - When statistics mode is Enabled - for each switched queue, record a counter for discarded packets, due to unavailable Buffer Descriptors.
  - When statistics mode is Disabled - for each switched queue, a counter for the total number of packets in the queue is available.
  - Record counters for number of transmitted/received packets per CID.
  - Record counters for number of transmitted/received ATM AAL2 cells per VC.
  - The switch operation is independent of CPS/SSSAR traffic.
  - Partial packet discard support for SSSAR traffic.
  - Partial Packet Received Storage (PPRS) Mode.
  - Weighted Fair Queueing (WFQ) mode.
  - WFQ Thresholds interrupts.
  - WFQ APC Enhancements.
- CPS Receiver
  - Segmentation of CPS PDU directly to external memory queues.

  - A separate queue for every VP | VC | CID or a common queue for multiple VP | VC | CID combinations.
  - Sequence number (SN) protection check for CPS-PDU.
  - CRC5 (HEC) check to detect errors in the CPS-PH of the CPS-Packet
  - OSF (offset field) of the STF (start of frame) check (a valid value is less than 48)
  - An SDU length limit parameter (Max_SDU_Deliver_Length) per ATM VC.
  - Odd parity check for the STF octet of the CPS-PDU.
  - Record counters for number of received packets per CID.
  - Record counters for number of received ATM AAL2 cells per VC.
- CPS Transmitter
  - Reassemble CPS PDU directly from external memory.
  - Perform CPS-PDU padding as needed.
  - Insert Sequence Number bit of the CPS-PDU.
  - Parity bit is calculated to provide odd parity over the STF octet.
  - Calculation of CRC-5 on the first 19 bits of the CPS-PH.
  - UUI field in the CPS-Packet header is programmed according to the value of the CPS_UUI parameter (per packet).
  - A free running counter (per TX Queue) is decremented for each packet sent.
  - Record counters for number of transmitted packets per CID.
  - Record counters for number of transmitted ATM AAL2 cells per VC.
- SSSAR Receiver
  - Reassemble CPS packets from the same CID into a SSSAR SDU.
  - A separate queue for every PHY | VP | VC | CID.
  - Perform all the above mentioned CPS receiver functions.
  - The UUI field is stored for the host into the RxBD after receiving the whole SSSAR frame.
  - A Ras_Timer mode is provided. When the Ras_Timer expires the buffer is closed with a timer expired error. The next packet received starts a new SSSAR SDU in a new buffer.
  - The SDU length is checked. If the frame exceeds the length limit (SSSAR_Max_SDU_Length), the receiver discards the rest of the packets from the current frame, closes the buffer and reports a Max_SDU violation error in the BD. The next packet received starts a new SSSAR SDU in a new buffer.
  - Partial Packet Discard. If no buffer is available when a packet arrives, the receiver enters a frame hunt state and discards each incoming packet from the current frame.
  - Record counters for number of received frames per CID.
  - Record counters for number of received ATM AAL2 cells per VC.
- SSSAR Transmitter
  - Segmentation of SSSAR SDUs from SSSAR TX queue into CPS packets.
  - An SSSAR TX queue may contain several CIDs.
  - Performs all the above mentioned CPS transmitter functions.

— A programmable segment length (Seg_Len) is copied from the SSSAR SDU into the CPS packet payload, except for when at the end of the frame or buffer.
— UUI mode available. The UUI value of the last packet in the SSSAR frame defaults to 26. When UUI mode is enabled, the SSSAR UUI is copied from the byte following the last byte of the frame.
— Record counters for number of transmitted frames per CID.
— Record counters for number of transmitted ATM AAL2 cells per VC.

41.3 AAL2 Transmitter

The following sections describe the AAL2 transmitter.

41.3.1 Transmitter Overview

A transmitter cycle starts when the APC schedules an ATM channel number for transmission. The TCT is fetched and the AAL type of the channel is checked. For AAL2 cells, the transmitter first handles uncompleted packets from the previous cell of the current CID (partial and split cases) by filling the beginning of the cell with the remainder of the last packet.

Then, the transmitter performs the priority mechanism (see Section 41.3.2, “Transmit Priority Mechanism”) in order to fill the cell with new packets. The priority mechanism determines the order in which the TX queues are serviced. The transmitter continues to search for ready packets in the TX Queues until either the cell is successfully filled with packets, or no more packets are ready but the cell is not yet completed. In the first case the cell is simply sent. In the latter case, the optional CNT_CU (described in Section 41.3.5.1, “AAL2 Transmit Connection Tables”) is examined. If the CNT_CU has expired, meaning that the same channel was scheduled CNT_CU plus one iterations and still the cell is uncompleted, the uncompleted cell is padded with zeros and sent; otherwise, the cell is temporarily stored in external memory for the RISC to attempt to complete it the next time the channel is scheduled.

The TX queues are the data structures that store the CPS packets and SSSAR frames. Each TX queue can contain different CIDs. Each TX queue is maintained by a Tx queue descriptor (TxQD) that holds the queue pointer and parameters to manage the queue.

When the transmitter fetches a packet out of an SSSAR TX Queue, it usually takes out of the SSSAR buffer a number of octets equal to TxQD[Seg_Len] (see Section 41.3.5.5, “SSSAR Tx Queue Descriptor). The channel CID is taken from the BD of the first buffer of the SSSAR frame (see Section 41.3.5.6, “SSSAR Transmit Buffer Descriptor”). A CPS UUI = 27 is used for all the in-frame packets until the last packet from the SSSAR frame is sent. The last packet can optionally contain a per frame, user-defined UUI. After an SSSAR buffer is completely sent, an optional interrupt event is issued to the host. Also, if an SSSAR TX queue is empty an optional interrupt event is issued to the host.

In case of CPS TX Queue, the transmitter fetches the packet header out of a Buffer Descriptor and the packet payload out of a CPS buffer (see Section 41.3.5.4, “CPS Buffer Structure”). The HEC in the Packet Header is calculated by the RISC. After a CPS packet is sent, an optional interrupt event is issued to the host. Also, if the CPS TX queue is empty an optional interrupt event is issued to the host.

The optional partial filled mode (see Section 41.3.3, “Partial Fill Mode (PFM)”) limits the number of data octets per cell. This can be used to ensure that a cell does not contain a split packet or to limit transmission to one packet per TX cell by setting a low partial fill threshold (PFT).

The no-STF (no start of frame) mode (see Section 41.3.4, “No STF Mode”) enables the transmission of cells that do not include the STF byte, thus allowing for 48-byte packets.

41.3.2 Transmit Priority Mechanism

The transmit priority mechanism operates in the following modes:

- Round robin (TCT[Fix]=0).
- Fixed priority (TCT[Fix]=1).
- Weighted Fair Queueing (WFQ) mode.

The following sections describe the priority options.

41.3.2.1 Round Robin Priority

In round robin priority mode, the Tx queues all have equal priority. The transmitter starts with the TxQD pointed to by TCT[FirstQueue], as shown in Figure 41-4. The number of packets that the transmitter services from each queue is determined by the one-packet bit (TCT[OneP]). If TCT[OneP]=0, the transmitter tries to process as many packets in the queue as needed to fill up the cell. Only when the queue is empty does the transmitter move on to the next queue (assuming the cell is not completed). If TCT[OneP]=1, the transmitter attempts to take only one packet out of each queue. (Set TCT[OneP] for implementations where each queue contains only one CID.)

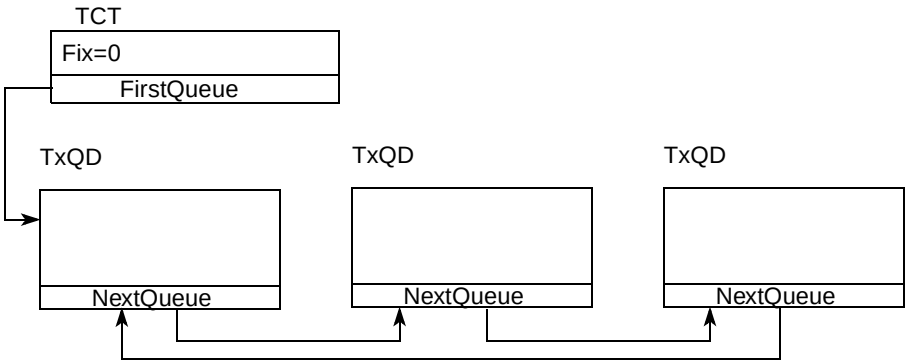


Figure 41-4. Round-Robin Priority

The transmitter steps from one TxQD to the next along the queue links. The TCT[MaxStep] parameter limits the number of TX Queues that the transmitter visits during a cell time. If MaxStep is reached before the cell has been completely filled, one of the following events takes place:

- TCT[ET]=0 (CNT CU disabled). The cell is padded with zeros and sent.
- TCT[ET]=1 (CNT CU enabled). If the Counter CU has not expired, the cell is not sent. (The transmitter attempts to fill the cell the next time this channel is scheduled.) If the Counter CU has expired, the cell is padded with zeros and sent.

After the transmitter sends a cell, it saves the queue link of the last TxQD serviced in TCT[FirstQueue].

41.3.2.2 Fixed Priority

In fixed priority mode (TCT[Fix]=1), the transmitter, with each new cell, starts with searching the highest priority queue and then moves on to the lower priority queues. The TCT[FirstQueue] points to the highest priority queue, as shown in Figure 41-5, and remains unchanged by the RISC.

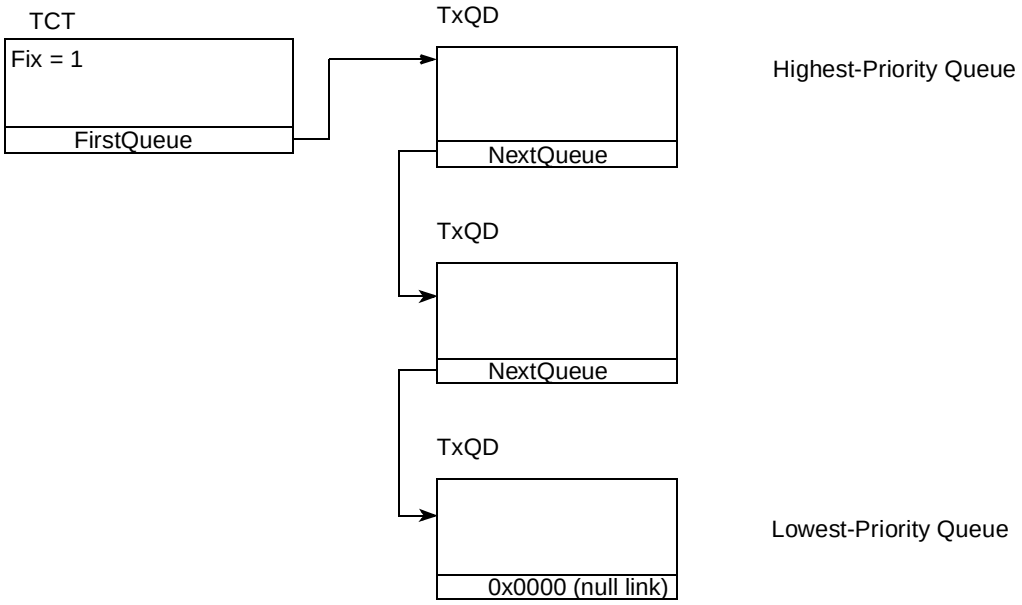


Figure 41-5. Fixed Priority Mode

The TCT[OneP] determines the number of packets that the transmitter attempts to take from each queue (see the explanation in round robin mode).

The NextQueue field of the lowest priority TxQD should be cleared (null link), and TCT[MaxStep] should be programmed to the total number of TX queues in the channel. When the transmitter reaches the null link and the cell is still not complete, the transmitter checks the TCT[ET] mode as described above for the round robin mode.

41.3.2.3 AAL2 Switch - Weighted Fair Queuing (WFQ) Prioritization

The Weighted Fair Queueing priority mode is enabled for an AAL2 channel by setting the TCT[WFQ_En] and RCT[WFQ_En] bits respectively. It is available for the AAL2 Switch mode only. This mode performs selection of up to four queues per channel, giving flexibility in scheduling and prioritization on a per queue basis.

The WFQ table contains a pointer to the TxQD base for this channel. An array of up to four TxQD is located in this memory location. These queues can be located in internal or external memory depending on the TCT[ExtTQD] bit. See Table 41-1.

This mode, when enabled, has enhanced queue management features which are important for switch mode. Another feature which could be invoked in this mode is scheduling which is based on the load of the system. The features are described in Section 41.4.3.1.1, “Queue Management” and in Section 41.4.3.1.2, “Automatic Bandwidth Allocation”.

Figure 41-6 describes the WFQ priority mode.

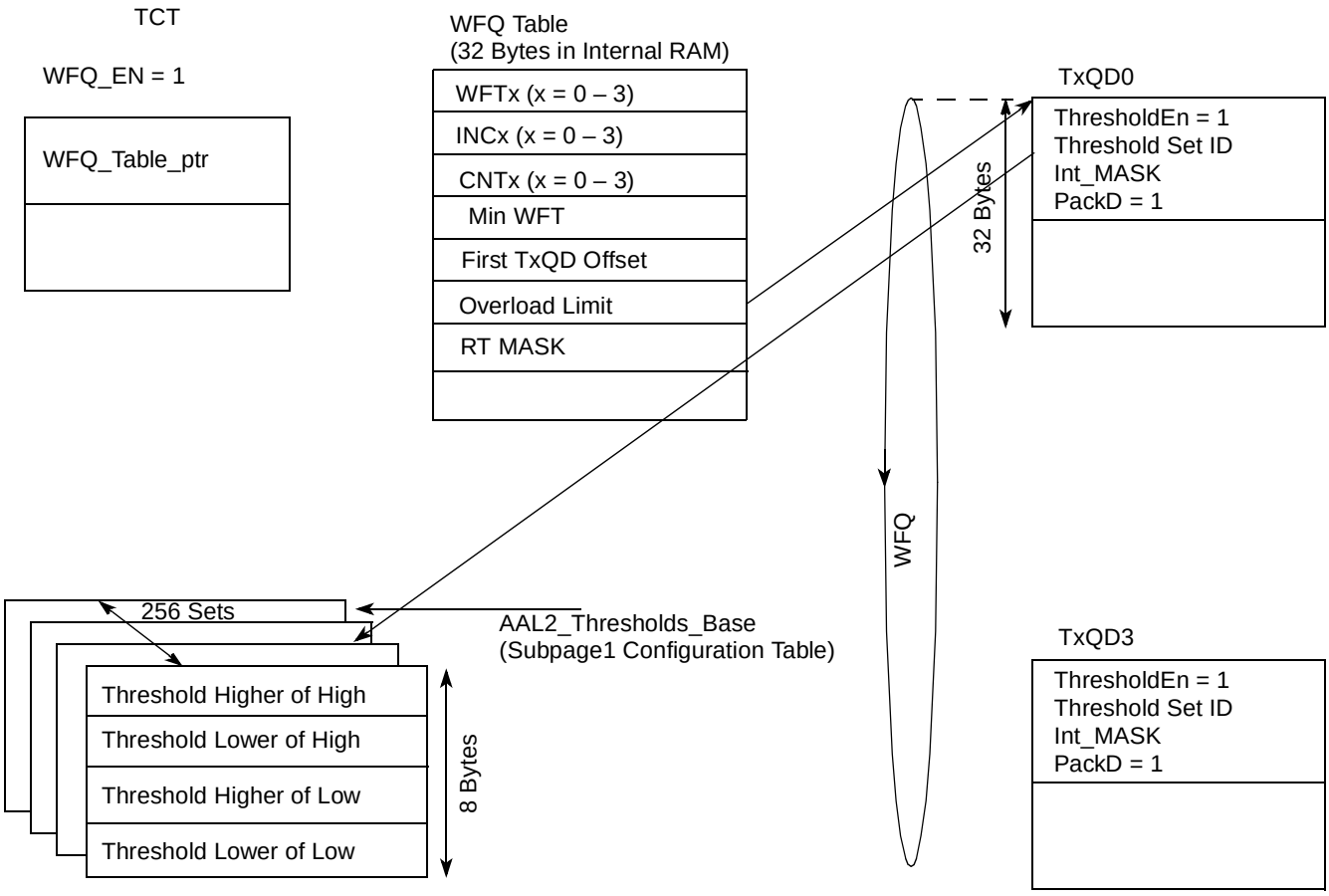


Figure 41-6. AAL2 WFQ Scheme

Each AAL2 channel supports up to four Tx queues (TxQDs) selected by the WFQ algorithm. These TxQDs are located consecutively in memory, each occupies 32 bytes. The first TxQD is pointed by the 'First_TxQD_offset' field in the WFQ Table. A queue is selected only if it has data ready for transmission. There is an automatic activation and deactivation of a queue by the receiver and transmitter respectively. This is done by setting and clearing the TxQD Queue Empty bit (QE) according to the number of packets in the queue which is maintained by the CNT entry of each queue.

When the transmitter has searched all the TxQDs and the cell is still not completed, the transmitter checks the TCT[ET] mode as described above for the round robin mode.

The WFQ table resides in the RAM and contains the management and parameters for the WFQ operation. (See Table 41-1) Each queue has a weight associated with it - WINC and scheduling is determined by this weight multiplied by each packet length.

NOTE

Since each weight is multiplied by the packet length factor, it is important that the weights would be as smaller as possible to improve performance (less WFT wrap scenario). Also the maximum value allowed for WINC is 511.

There are two modes of operation for queue selections. This is determined by the TCT[OneP] bit:

1. OneP mode is enabled - the WFQ selection is done for each packet in the cell.
2. OneP mode is disabled - the WFQ selection is done at the beginning of a cell (or after part condition), and the selected TxQD remains valid as long as there is data ready for transmission in this queue. Once there are no more packets to transmit, a new selection is done.

The formula for the setting the WINC value is:

i=0..3

TxQD#i packet rate = highest packet rate* (1/INCi)/[1/INC1+1/INC2+1/INC3+1/INC4]

41.3.2.3.1 WFQ Table

Under each TCT up to 4 TxQDs are available, therefore 4 weighted values are needed. The WFQ table occupies 32 bytes in RAM, and it is accessed by a pointer from the TCT[WFQ_Table_ptr].

Table 41-1. WFQ Table

Offset	Bits	Name	Width	Description
0x0-0x7	—	**WFTi (i=0..3)**	Hword *4	Weighted Finish Time. Eight 16 bit WFTs, one for each TxQD. The WFT is used to determine the finish time of the next packet being transmitted from the queue. WFT at offset zero corresponds to TxQD0, WFT at offset 0x06 corresponds to TxQD3. see Initialization instructions in Table 41-2
0x8-0xf	—	**WINC (i=0..3)**	Hword *4	Weighted Increment value. See Table 41-3.
0x10-0x17	—	CNTi (i=0..3)	Hword *4	TxQD Counter. Counts the number of packets in each TxQD. Should be initialized to 0.
0x18	—	**First TxQD Offset**	Hword	TxQD Base Address Offset. User initialized to the base address of the first TxQD entry. Could be internal or external memory address. Aligned to 32 bytes. In case TCT[ExtTQD]=1 this entry acts as an offset from TxQDExtBase address multiplied by 32. See Section 41.3.5.2, “External TxQD versus Internal TxQD”. **Note:** The user must allocate the TxQDs, if they resides in external memory, so that each TxQD occupies 32 bytes of memory, even that the actual TxQD size is 16 bytes.
0x1A	—	MinWFT	Hword	Initialized to 0. Used by the QUICC Engine block.

Table 41-1. WFQ Table (continued)

Offset	Bits	Name	Width	Description
0x1C	—	**Overload_Limit**	Hword	Overload Limit (in slot number units). When the number programmed in this field is zero this mode is **disabled**. APC scheduling: Overload Limit is given in slot number units. The maximum APC slot difference between [RPTR-SPTR] mod APC slot number, that behind it the APC would not send any best effort (BE) cell (assuming no RT cells are available). See Section 41.4.3.1.2, "Automatic Bandwidth Allocation". **Note:** If APC is working in scalable mode, and the Overload Limit mode is enabled, then the user must set the Ext_CS bit in the control slot, and allocate additional 4-6 bytes after the control slot, see Figure 32-53. GCRA scheduling: Overload Limit is given in time stamp units. The maximum time between the system time and the channel finish time in time stamp units, that behind it the APC would not send any best effort (BE) cell (assuming no RT cells are available), see Section 32.2.12, "GCRA Scheduler."
0x1E	0-3	**RT MASK**	—	Real-Time traffic Mask. Bits 0-3 identify which of the TxQDs are Real Time traffic (RT- masked by ones) and which of the queues are Best Effort traffic (BE masked by zeros). e.g. 0xc0- means that TxQD0 and TxQD1 are RT, and TxQD 3 and TxQD4 are BE. Valid only if Overload_limit is bigger than 0.
	4-7	RT MASK TMP	—	Should be cleared. Used by the QUICC Engine block.
	8-15	—	Byte	Reserved, should be cleared.

41.3.2.3.2 WFTi

Figure 41-7 shows the format of the Weighted Fair Queueing (WFQ) Finish Time entry. There are four such entries in the table, one for each queue. These entries are initialized by the CPU, and are not accessed by it when the switching function is enabled.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QE	**FTM**													**QID**	

Figure 41-7. WFT Entry

Table 41-2 describes the WFT fields.

Table 41-2. WFT Field Descriptions

Bits	Name	Description
0	QE	TxQD Queue Empty. **Initialized to 1**. This will disable the transmission from this TxQD until the event of receiving a packet to this TxQD. **Note:** This bit is set automatically by the QUICC Engine block if the corresponding TxQD has no packets to transmit under its TxQD, and on the other hand this bit is cleared on the event of receiving a packet to this queue.
1-13	FTM	Finish time. This bit is automatically updated by the QUICC Engine block. **Initialize with zero**.
14-15	QID	TxQD ID. The CPU initializes the TxQD ID number in this field. This will be used as the index to the TxQD structure. Note that the QID is also part of the comparison algorithm, so if there is a case of 2 TxQDs with the same Finish Time values, the lowest TxQD will be selected.

41.3.2.3.3 INCi

Figure 41-8 shows the format of the Weighted Fair Queueing (WFQ) Increment entry. There are four such entries, one in each TxQD.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
—	INC													Reserved	

Figure 41-8. WINC Entry

Table 41-3 describes the WINC fields.

Table 41-3. WINC Field Descriptions

Bits	Name	Description
0		Reserved. Initialize to zero.
1–13	INC	Weighted Increment value. This value determines the relative bit rate for the TxQD. It is multiplied by each transmitted packet length in order to determine the next FTM value.
14–15		Reserved. Initialize to zero.

NOTE

The WFQ algorithm can implement Strict/Fixed Priority by setting INC=0 for all queues, or as round robin, by setting equal INC values for all queues.

41.3.3 Partial Fill Mode (PFM)

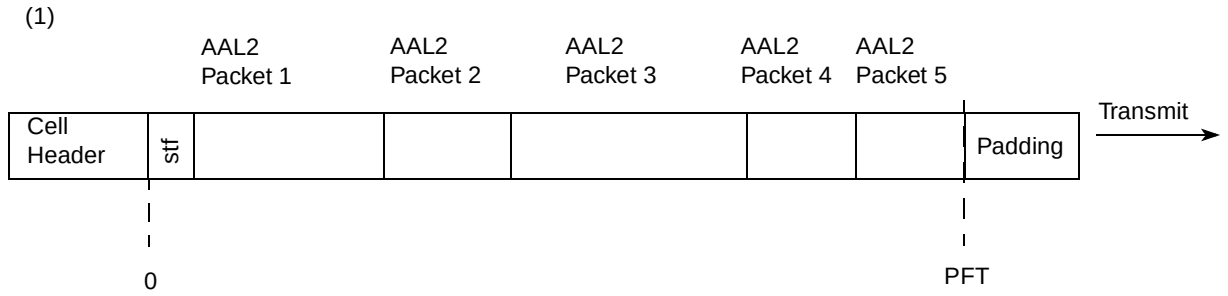
The partial fill mode (TCT[PFM]=1) allows the user to specify a partial fill threshold (TCT[PFT]), which limits the number of data octets sent with each ATM cell. Partial fill mode assures that packets are not split over two cells, unless the first packet of the cell is greater than 47 bytes.

In partial fill mode, the transmitter starts by filling the ATM cell with the first packet. After the first packet, the QUICC Engine block determines if including the next packet in the cell would exceed the PFT limit.

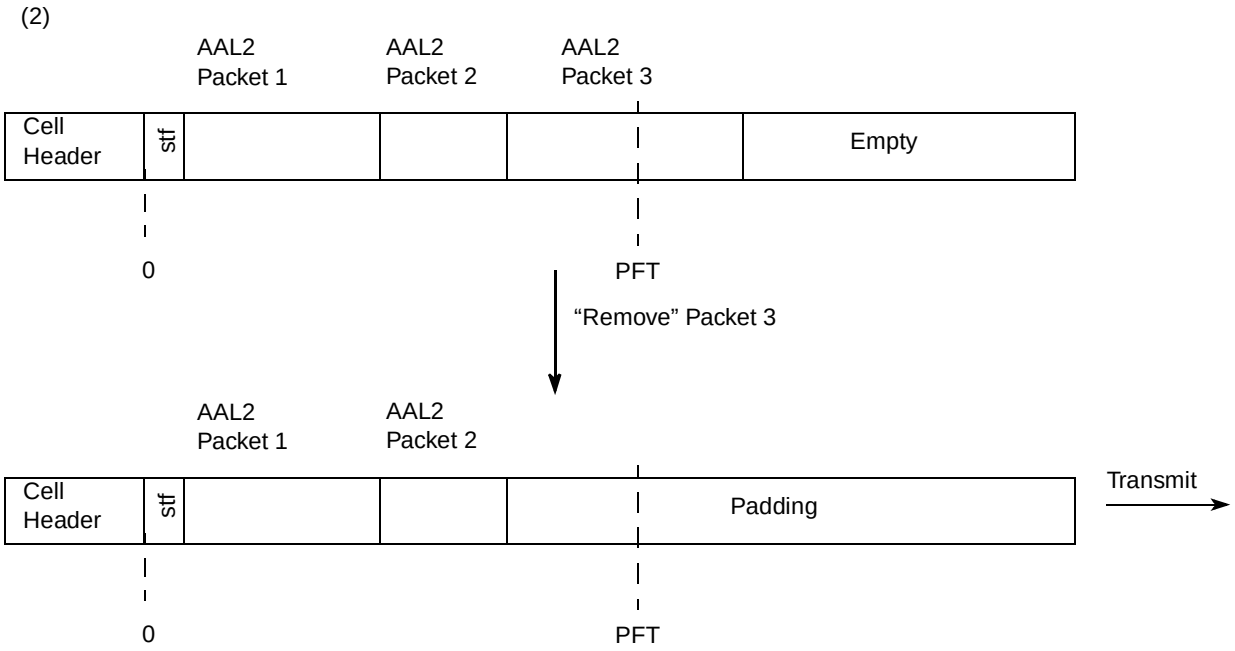
If so, the second packet is not inserted into the current cell, the unused payload is padded with zeros, and the cell is sent. If the second packet does not exceed PFT, the QUICC Engine block inserts it into the CPS_PDU and moves on to the third packet, and so on.

By programming PFT = 1, the partial fill mode can also serve to assure that only one packet is sent per cell.

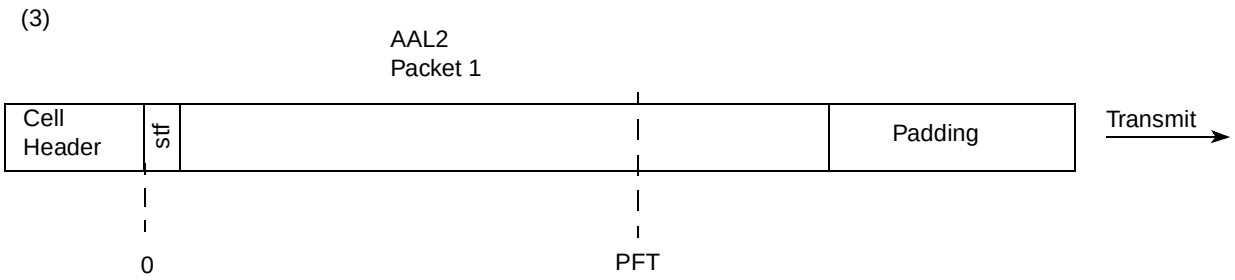
The following figures provide examples of partial fill mode operation:



In example (1), five packets fit exactly within the PFT limit.



In example (2), because PFT is less than the combined lengths of packet1, packet2 and packet3, the ATM cell is sent only with packets 1 and 2. (Packet3 will be sent with the next cell.)



In example (3), because the first packet exceeds the PFT value, the CPS-PDU consists only of this packet (and the unused octets are padded with zeros).

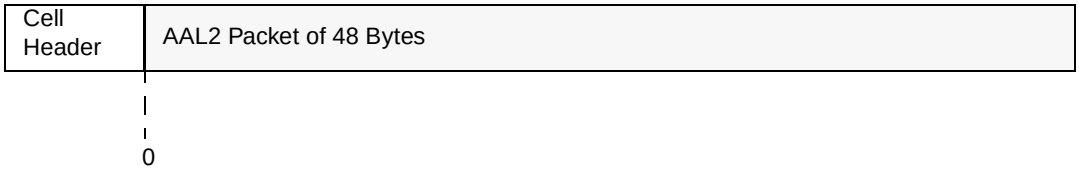
41.3.4 No STF Mode

The no-STF (no start of frame) mode enables the transmission of 48-byte packets by not including the STF byte in the CPS PDU. The no-STF mode should always be used with PFT programmed to 48 in order to prevent split and partial packets. For the AAL2 channels that use this mode, packets must not be larger than a maximum packet size of 48.

The user activates this mode per ATM channel by:

- setting the TCT[NoSTF] bit,
- and establishing a partial fill threshold by programming TCT[PFM]=1 and TCT[PFT]=48.

The following figure shows a cell using no-STF mode:



41.3.5 AAL2 Tx Data Structures

The following sections describe the TCT and the structures in which CPS packets and SSSAR SDUs are stored in memory.

41.3.5.1 AAL2 Transmit Connection Tables

The transmit connection table (TCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the TX Queues of the ATM channel are maintained here. Figure 41-9 shows the AAL2-specific TCT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		CETM	DTB	BDB	AVCF	PFM	ATT		ET	VCON	INTQ	
Offset + 0x02	—	—	—	—	—	—	—	—	—	—	—	—	NoSTF	AAL		
Offset + 0x04	—															
Offset + 0x06																
Offset + 0x08	—								—							
Offset + 0x0A	FirstQueue/FirstQD_Offset/WFQ_Table_ptr															
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10	CNT_CU								CNT_CU_Shadow							
Offset + 0x12	—															
Offset + 0x14	—															
Offset + 0x16	APC Linked Channel / GCRA Burst Tolerance															
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)															
Offset + 0x1a																
Offset + 0x1C	—		PMT						MaxStep							
Offset + 0x1E	WFQ_En	StatsIE	PFT						ExtTQD	—	ExtStats Addr	Stats En	OneP	STPT	Fix	PM

Figure 41-9. AAL2 TCT

NOTE

When the channel is active, the QUICC Engine block fetches the TCT (32 bytes) using burst cycle and writes back only the first (24 bytes).

Table 41-4 describes the AAL2 TCT fields.

Table 41-4. AAL2 TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "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 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 19-1 and Section 5.4.3.7, "Debug Configuration."
	6	**DTB**	Data buffer and UDC table bus selection. 0 Data buffers and UDC table reside on the coherent system bus. 1 Data buffers and UDC table reside on the secondary bus.
	7	**BDB**	Bus selection for the BDs, interrupt queues, the free buffer pool, etc. TxQD and the CID Statistics Tables (Only if they resides in external memory - TCT[ExtStatsAddr]=1). 0 Reside on the coherent system bus. 1 Reside on the secondary bus.
	8	**AVCF**	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table. 0 The APC does not remove this VC from the scheduling table and continues to schedule it to transmit. 1 The APC removes this VC from the scheduling table.Functionality of the APC varies depending on other setting in the system. If the TCT[VCON] bit is set, indicating that the automatic activation for switch mode is disabled, the QUICC Engine block removes the VC from the scheduling table. It clears the VCON bit indicating removal of the channel. In order to resume transmission the host has to initiate a new ATM TRANSMIT command is needed. If the TCT[VCON] bit is cleared, indicating that the automatic activation for switch mode is enabled, the QUICC Engine block removes the VC from the scheduling table. Transmission is resumed automatically by the receiver with no need for any host intervention. See Section 41.4.3, "AAL2 Switching" for more description.

Table 41-4. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	9	**PFM**	Partial fill mode. See Section 41.3.3, "Partial Fill Mode (PFM)." 0 Partially filled cells are not supported. 1 Partially filled cells are supported.
	10–11	**ATT**	ATM traffic type 00 Peak cell-rate pacing (regular traffic). The host must initialize PCR and PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and Minimum cell rate pacing. The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 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 32.3.4.3, "GBR Programming Model," on page 32-94.
	12	**ET**	Enable CNT_CU. 0 CNT_CU operation in this channel is disabled. 1 CNT_CU operation in this channel is enabled.
	13	**VCON (status)**	Virtual channel is on. Should be set by the host before it issues an ATM TRANSMIT command. When the host sets the STPT (stop transmit) bit, the QUICC Engine block deactivates this channel and clears VCON. The same applies When the TCT[AVCF] is set. The host activates this channel by an ATM TRANSMIT command only when the QUICC Engine block clears VCON.
		VCON (mode)	Cleared by the host to enable the automatic activation in switch mode - AVCON/AVCF mechanism. When this mode is enabled, the TCT[AVCF] bit should be set. See Section 41.4.3, "AAL2 Switching" for more description.
	14–15	**INTQ**	Points to one of the four interrupt queues available.
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	**NoSTF**	No STF byte. See Section 41.3.4, "No STF Mode." 0 Normal AAL2 cell structure. 1 The cell does not include the STF byte. In this mode each cell starts with a new packet and contains only whole packets (no split or partial).
	13–15	**AAL**	AAL type 000 AAL0 —Segmentation with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol. (Use this configuration for rev A parts only, unless needed for backwards compatibility to existing software development.) 010 AAL5 —ATM adaptation layer 5 protocol 011 AAL3/4 —ATM adaptation layer 3/4 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1 with CES —ATM adaptation layer 1 protocol with optional support for circuit emulation service. (For rev B and later parts, this configuration is strongly recommended for all AAL1 applications, with or without CES.) All others reserved.
0x04		—	Reserved, should be cleared during initialization.
0x06		—	Reserved, should be cleared during initialization.
0x08		—	Reserved, should be cleared during initialization.

Table 41-4. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x0A		**FirstQueue / FirstQD_Offset /** **WFQ_Table_ptr**	If **WFQ_En** bit is cleared: Points to the first queue to be serviced in the transmitter cycle. In round-robin priority mode (TCT[Fix]=0), this pointer could point to any one of the TxQDs related to this channel. In fixed priority mode (TCT[Fix]=1), this pointer should point to the highest priority TxQD. In case TCT[ExtTQD]=1 this entry acts as an offset from TxQDExtBase address multiplied by 32. See Section 41.3.5.2, “External TxQD versus Internal TxQD”. If **WFQ_En** bit is set: WFQ Table pointer. Allocates 32 bytes for the WFQ management. Should be 8 bytes aligned. See Table 41-1.
0x0C	0–7	**Rate Remainder**	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared during initialization by the user.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot for APC Mode/ or in CETSCR 1/256 time units in case of GCRA schedule.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots for APC Mode/ or in CETSCR time unit in case of GCRA scheduler) permitted for this channel according to the traffic contract.
0x10	0–7	**CNT_CU**	Count CU. Valid only if TCT[ET] is set. If the channel was scheduled CNT_CU plus one iterations and still the cell is uncompleted, the uncompleted cell is padded with zeros and sent; otherwise it is temporary stored in external memory till the next scheduling. Assures that the CPS-packet already packed in a cell wait at most the average channel period multiplied by the ‘CNT_CU-1’.
	8–15	**CNT_CU_Shadow**	Count CU Shadow. Valid only if TCT[ET] is set. This field must be initialized to the CNT_CU value.
0x12	—	Res	Reserved, should be cleared during initialization.
0x14	—	Res	Reserved, should be cleared during initialization.
0x16	—	**APCLC/** **GCRA_BT**	**APC Mode:** APC linked channel. Used by the QUICC Engine block. Should be cleared during initialization. **GCRA Scheduler:** When working in GCRA scheduler mode, this field should be initialized to the Burst tolerance Limit of this channel in the TSR time unit, see Section 32.2.12.1, “GCRA Scheduler Rate Programming.”
0x18	—	**ATMCH**	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PMT**	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT*32.
	8–15	**MaxStep**	Valid only if WFQ_En bit is cleared. Holds the number of TX Queues visited for each cell being prepared for transmission. In fixed priority mode (TCT[Fix]=1), MaxStep should be set to the total number of TX queues in the channel. See Section 41.3.2, “Transmit Priority Mechanism.”

Table 41-4. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1E	0	**WFQ_En**	Weighted Fair Queueing Enable. 0 - WFQ priority mode is disabled. 1 - WFQ priority mode is enabled. See Section 41.3.2.3, “AAL2 Switch - Weighted Fair Queuing (WFQ) Prioritization”. **Note:** All TxQDs under a TCT with WFQ_En bit set, must be AAL2 switch TxQDs.
	1	**StatsIE**	Statistics Interrupt Enable. 0 - Tx CID Statistics / ATM cell counters overflow interrupt is disabled. 1 - Tx CID Statistics / ATM cell counters overflow interrupt is enabled. SeeSection 41.3.5.7, “Transmitter Statistics Structure”.
	2–7	**PFT**	Partial fill threshold. Used for partially filled cells only; see Section 41.3.3, “Partial Fill Mode (PFM).” Specifies the maximum number of packet bytes allowed in a CPS PDU. The range 1–48 are valid values. If PFT=48 in partial fill mode, performance is adversely affected. When not in partial fill mode, PFT must be initialized to 47.
	8	**ExtTQD**	External TxQD. 0- TxQDs which belong to this TCT, are located in internal RAM. See Section 41.3.5.2, “External TxQD versus Internal TxQD”. 1- TxQDs which belong to this TCT, are located in external RAM. See Section 41.3.5.2, “External TxQD versus Internal TxQD”.
	9	**—**	Reserved, should be cleared.
	10	**ExtStats Addr**	Using External TxCIDStatsAddress Table or internal multi-user RAM TxCIDStatsOffset Table for getting the TxCID Statistics Table Entry location. See Section 41.3.5.7, “Transmitter Statistics Structure”. 0- TxCIDStatsOffset Table and the TxCID Statistics Table itself resides in internal multi-user RAM. The Offset in the TxCIDStatsOffset Table is the offset from internal RAM to the TxCID Statistics Table (which can consist up to 256 2 bytes CID counters) - which belongs to this Tx channel. 1- TxCIDStatsAddress Table and the TxCID Statistics Table itself resides in external memory. The Address in the TxCIDStatsAddress Table, which is of 4 bytes length, is the direct external address to the TxCID Statistics Table entry (which can consist up to 256 2 bytes CID counters) - which belongs to this Tx channel.
	11	**StatsEn**	Statistics mode Enable bit. 0- CID and Cell counters are disabled for this VC. 1- CID and Cell counters are enabled for this VC. See Section 41.3.5.7, “Transmitter Statistics Structure”.
0x1E	12	**OneP**	One packet per queue. Valid also when WFQ_En is set. See Section 41.3.2, “Transmit Priority Mechanism.” 0 The transmitter reads as many packets as possible from each TX queue before moving to the next queue. 1 The transmitter reads only one packet from each TX queue before advancing to the next queue.

Table 41-4. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	13	**STPT**	Stop transmit. Should be cleared during initialization. This bit should not be used when AVCON/AVCF mode is enabled (other than debug purposes). When the host sets this bit, the QUICC Engine block removes this channel from the APC and clears TCT[VCON] flag. **Note:** 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
	14	**Fix**	Fixed priority. Valid only if WFQ_En bit is cleared. See Section 41.3.2, "Transmit Priority Mechanism." 0 Round robin priority. The TX Queues related to this channel all share the same priority. 1 Fixed priority. The TX Queues are ordered in a fixed priority ladder.
	15	**PM**	Performance monitoring 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.
NOTE: Entries in boldface must be initialized by the user.			

41.3.5.2 External TxQD versus Internal TxQD

There is an option to use External TxQDs together with Internal TxQDs, in order to support up to 64K TxQDs. This mode is enabled, per channel (TCT), by setting TCT[ExtTQD]. The QUICC Engine block fetches the external TxQD by adding to the TxQDExtBase, see Table , the FirstQD_Offset/NextQD_Offset multiplied by 32. The TxQDExtBase and each TxQD located in external memory should be located in 32 bytes aligned address. Internal TxQDs should be located in address which is aligned to their size (CPS TxQD - 16 bytes, and Switch TxQD/ SSSAR TxQD 32 bytes). See Figure 41-10 for description of the External TxQDs Structure.

Selection of the Bus is according to TCT[BDB] bit.

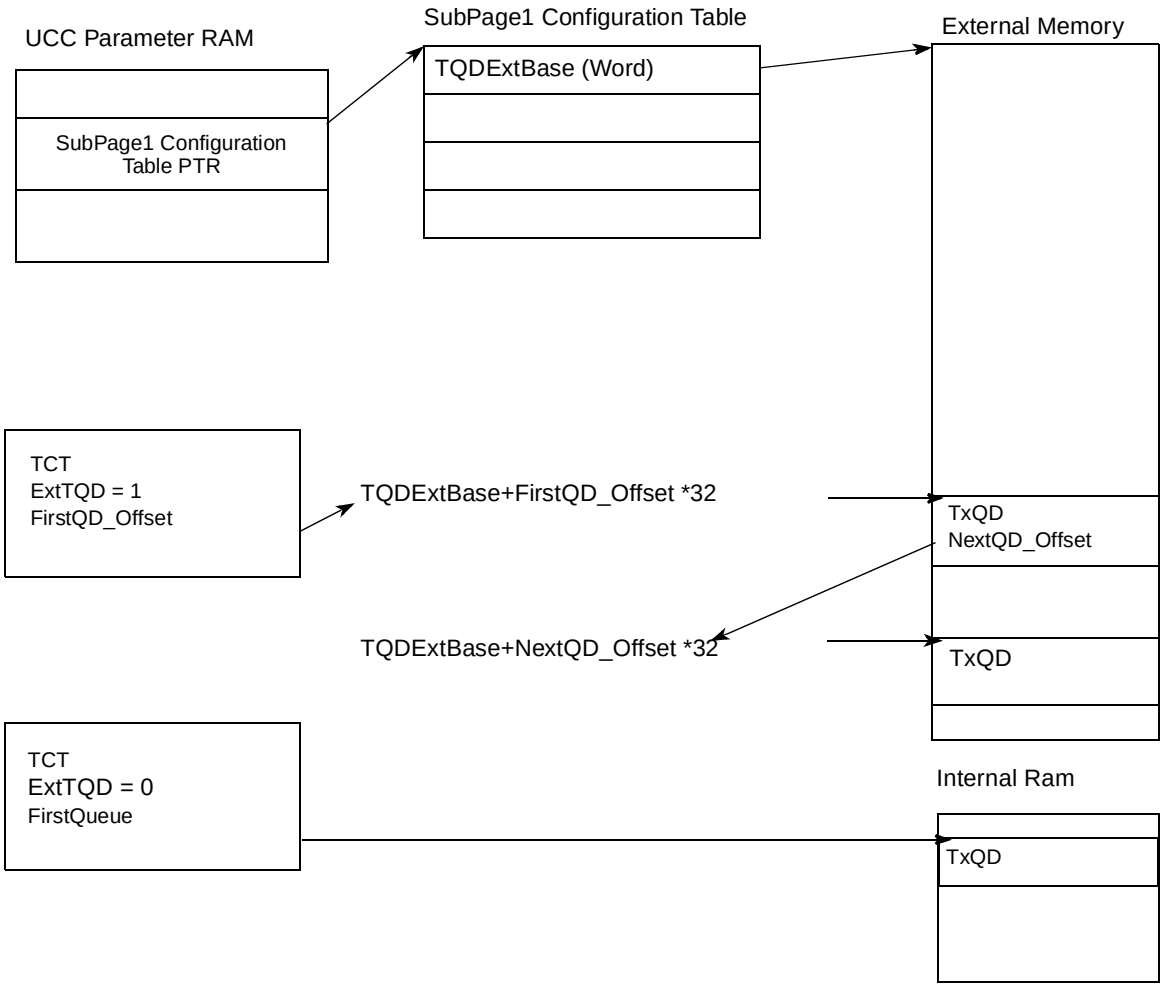


Figure 41-10. External TxQD Structure

41.3.5.3 CPS Tx Queue Descriptor

Each CPS TxBD table is managed by a CPS Tx queue descriptor (TxQD), as shown in Figure 41-11. The TxQD contains the address of the next BD to be serviced, and other queue-specific parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 41.3.2, "Transmit Priority Mechanism." The CPS TxQD could be located in the multi-user RAM, or in external memory, and it consumes 16 bytes for non Switch CPS and 32 bytes for Switch CPS.

	0	1	2	3	4 5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—	—	—	—	—	—	—	**BNM**	**SW**	**HEC**	**CPS**	**TBM**	—	—	**PackD**
Offset + 0x02	**TxBD Table Offset In** (switched mode only)														
Offset + 0x04	**TxBD Table Base**														
Offset + 0x06															
Offset + 0x08	**TxBD Table Offset Out**														
Offset + 0x0A	**Number of Packets In Queue/Discarded Packets**														
Offset + 0x0C	**NextQueue/NextQD_Offset**														
Offset + 0x0E	—							—							
Offset + 0x10	**SIZE** (switched mode only)														
Offset + 0x12	**Queue Management** (switched mode only)														
Offset + 0x14- Offset + 0x1F	—														

Figure 41-11. CPS Tx Queue Descriptor (TxQD)

Table 41-5 describes the CPS TxQD fields.

NOTE

When working in switch mode Type-1, and TxQD[PackD] bit is cleared - a number-of-packets-in-queue counter is available in the TxQD. The QUICC Engine block increments the counter for each packet received and decrements it for each packet sent. The host can poll the counter periodically to verify that the switching queues are not over-loaded.

With TxQD[PackD] bit is set, and working in switch mode (Type-1 or Type-2)- a number_of_discarded_packets counter is available in the TxQD. The QUICC Engine block increments this counter whenever a received packet is discarded due to unavailable buffers. When there is Overflow in number_of_discarded_packets counter, then a PackDOF event is issued in the interrupt queue. See Section 41.8.1, “Interrupt Entry for CID Statistics.”

When working in switch mode using Type-2 (see Figure 41-29), TxQD[PackD] bit must be set.

.

Table 41-5. CPS TxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–7	—	Reserved for internal use. (Used to save the BD status of the open BD.)
	8	**BNM**	Buffer not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**SW**	Switching queue. 0 Normal TX Queue. 1 This TxQD handles a switching queue. The receiver and transmitter share this queue.
	10	**HEC**	HEC calculation. 0 Transmitter calculates the CPS header HEC. 1 The CPS header HEC is taken from the CPS buffer descriptor.
	11	**CPS**	Sublayer type. For a CPS TxQD, this field must be set. 0 SSSAR. 1 CPS packet.
	12	**TBM**	Transmit buffer interrupt mask. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue). 1 The transmit buffer event of this queue is enabled.
	13–14	—	Reserved, should be cleared during initialization.
	15	**PackD**	Packet Discard. This bit is relevant ONLY for Switch mode. For External TxQD this bit MUST be set to 1. 0 - Count number of packets in Queue (Only for Switch Type-1). 1 - Count number of discarded packets in Switched queue. When TCT[WFQ_En] is set then PackD should be set, since the number of packets in queue is available in the WFQ table (see Table 41-1).
0x02	—	**TxBD Table Offset In**	Used only when this queue is used for switching (SW=1). Should be cleared during initialization.
0x04	—	**TxBD Table Base**	This pointer points to the base address of the BD table. Should be 8 byte aligned.
0x08	—	**TxBD Table Offset Out**	Holds the offset from the TxBD table base to the next BD to be opened by the transmitter. Should be cleared during initialization.
0x0A	—	**Number of Packets In Queue/Discarded Packets**	It could have two different meanings: When TxQD[PackD]=0: ->Number of Packets in Queue This field is NOT valid when working with Switch Type-2. Counts the number of packets currently in the queue. If this queue is switch Type-1, then the receiver increments this counter with each new received packet and the transmitter decrements it with each packet sent. For switch Type-1, the user should initialize this counter to zero. When this queue is not switched, this counter counts down with every packet sent. (This can have various purposes such as evaluating the packet rate that is transmitted from this queue.). When TxQD[PackD]=1: -> Discarded packets. Relevant Only for Switched TxQD. Count the discarded packets for this Switched TxQD due to unavailable Buffer Descriptors (buffer BSY condition).

Table 41-5. CPS TxQD Field Descriptions (continued)

Offset	Bits	Name	Description
0x0C	—	**NextQueue/ NextQD_Offset**	**Valid only when TCT[WFQ_En] is cleared:** Points to the next TxQD to be serviced after this one. See Section 41.3.2, "Transmit Priority Mechanism." In case TCT[ExtTQD]=1 this entry acts as an offset from TxQDExtBase address multiplied by 32. See Section 41.3.5.2, "External TxQD versus Internal TxQD".
0x10	—	**SIZE**	**Valid For switch mode only.** Size of BD Ring, in byte units. e.g., if BD ring consists of n BDs then set this value to n*8.
0x12	—	**Queue Management**	**Valid only when TCT[WFQ_En] is set.** See details in Figure 41-12.
0x14–0x1F	—		Reserved. Should be cleared.

NOTE: Entries in boldface must be initialized by the user.

Figure 41-12 describes the Queue Management Field, which is valid for WFQ mode only.

Offset	0	1	2	3	4	5	6	7	8 9 10 11 12 13 14 15
0x12	**Thr_En**	—	**Full Speed to Decelerate interrupt Mask**	**Decelerate to Full Stop interrupt Mask**	**Accelerate to Decelerate interrupt Mask**	**Full stop to Accelerate interrupt**	**Accelerate to Full Speed interrupt**	**Decelerate to Accelerate interrupt**	**Thr_Set_ID**

Figure 41-12. Queue Management Field

Table 41-6. Queue Management Field description

Offset	Bits	Name	Description
0x12	0	**Thr_En**	**Valid only when TCT[WFQ_En] is set:** Threshold Enabled. Valid only when working in WFQ Mode (TCT[WFQ_En]=1). 0- Threshold mechanism is disabled. 1- Threshold mechanism is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	1	**—**	reserved. Should be cleared.
	2	**Full Speed to Decelerate interrupt Mask**	**Valid only when TCT[WFQ_En] is set:** Full Speed to Decelerate interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	3	**Decelerate to Full Stop interrupt Mask**	**Valid only when TCT[WFQ_En] is set:** Decelerate to Full Stop interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	4	**Accelerate to Decelerate interrupt Mask**	**Valid only when TCT[WFQ_En] is set:** Accelerate to Decelerate interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	5	**Full stop to Accelerate interrupt**	**Valid only when TCT[WFQ_En] is set:** Full Stop to Accelerate interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	6	**Accelerate to Full Speed interrupt**	**Valid only when TCT[WFQ_En] is set:** Accelerate to Full Speed interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	7	**Decelerate to Accelerate interrupt**	**Valid only when TCT[WFQ_En] is set:** Decelerate to Accelerate interrupt Mask. 0 - this interrupt is disabled. 1 - this interrupt is enabled. For more description see Section 41.4.3.1.1, “Queue Management”.
	8–15	**Thr_Set_ID**	Threshold Set ID. Selects one of the available thresholds sets (0-255). This field can be changed dynamically by the user, by writing to directly to the **byte** containing these bits. For more description see Section 41.4.3.1.1, “Queue Management”.

NOTE: Entries in boldface must be initialized by the user.

41.3.5.4 CPS Buffer Structure

The CPS buffer structure consists of a BD table that points to data buffers. The BDs contain, apart from the buffer pointer, also the packet header. The buffers contain the packet payload. See Figure 41-13.

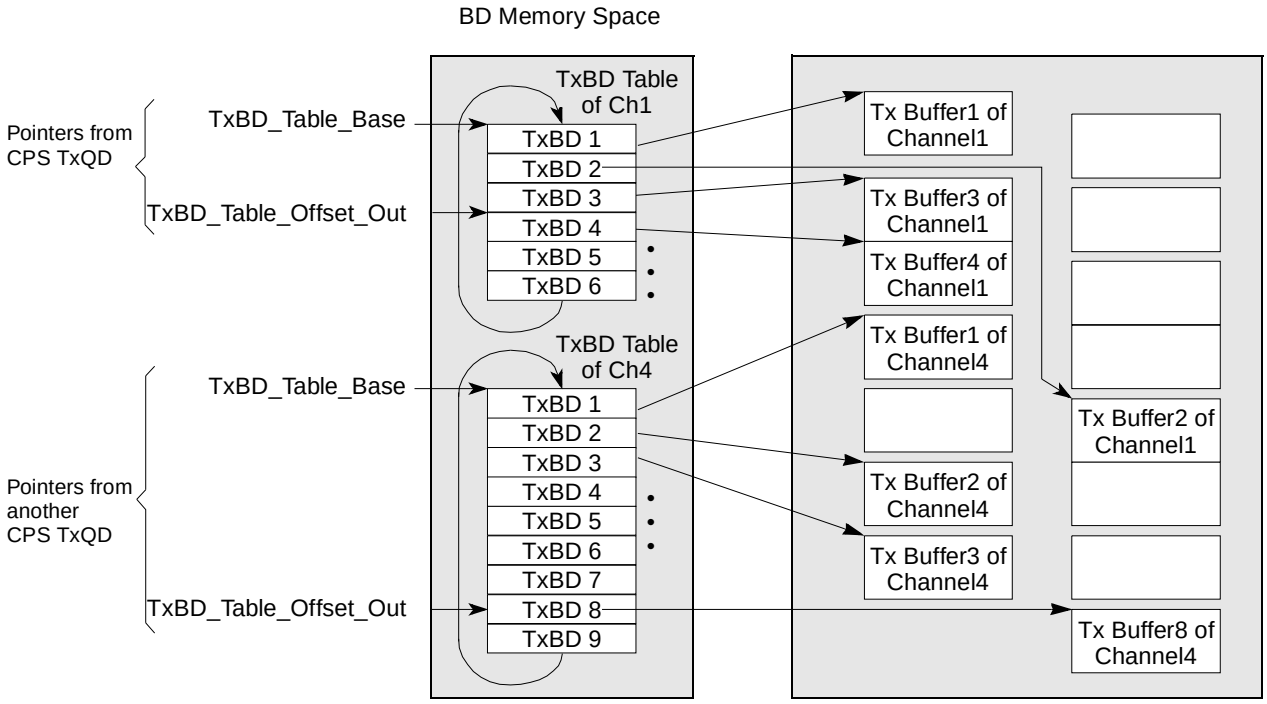


Figure 41-13. Buffer Structure Example for CPS Packets

Figure 41-14 shows a CPS TxBD.

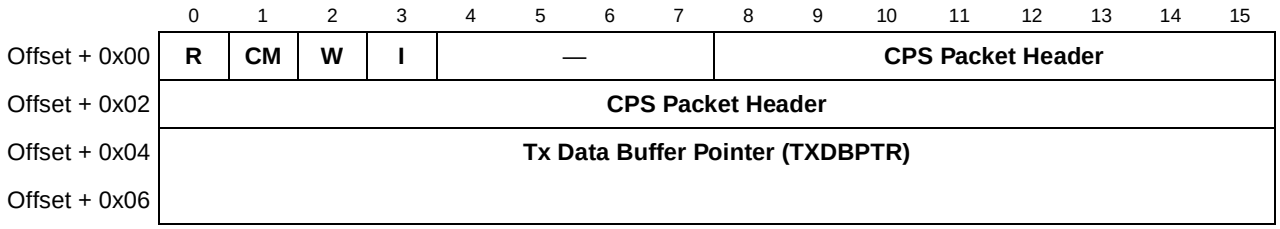


Figure 41-14. CPS TxBD

Table 41-7 describes the CPS TxBD fields.

Table 41-7. CPS TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**R**[1]	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	**CM**[2]	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 the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the QUICC Engine block transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4–7	**—**	Reserved, should be cleared during initialization.
	8–15	**CPS Packet Header**	This field contains the beginning (MSB) of the 3-byte packet header. See Figure 41-15 for the CPS packet header format.
0x02		**CPS Packet Header**	This field contains the rest of the packet header. If TxQD[HEC] = 0, the HEC part of the packet header is calculated by the QUICC Engine block, and the user may disregard the five least-significant bits of this field. See Figure 41-15 for the CPS packet header format.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This pointer is not modified by the QUICC Engine block.

NOTE:

1 Entries in boldface must be initialized by the user.

2 Setting Continuous mode (TxBD[CM] = 1) is not allowed in CID switching mode.

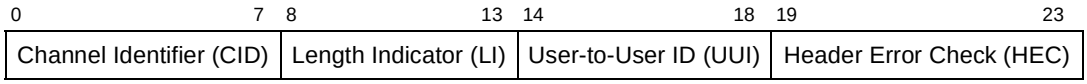


Figure 41-15. CPS Packet Header Format

41.3.5.5 SSSAR Tx Queue Descriptor

The SSSAR TX Queue Descriptor is described in Figure 41-16. The TxQD contains the base address of the BD table, the offset of the next BD to be serviced, the data buffer pointer, and other queue-specific

parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 41.3.2, “Transmit Priority Mechanism.” The SSSAR TxQD is located in the multi-user RAM or in external memory in a 32-byte aligned address.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—	—					—	—	BNM	UUI	INF	CPS	TBM	SSSAR	—	
Offset + 0x02	Seg_Len								—							
Offset + 0x04	TxBD Table Base															
Offset + 0x06																
Offset + 0x08	TxBD Table Offset Out															
Offset + 0x0A	—															
Offset + 0x0C	NextQueue/NextQD_Offset															
Offset + 0x0E	—															
Offset + 0x10	—															
Offset + 0x12																
Offset + 0x14	—															
Offset + 0x16																
Offset + 0x18	—															
Offset + 0x1a																
Offset + 0x1C	—															
Offset + 0x1E	—	—	—	—	—	—			—							

Figure 41-16. SSSAR Tx Queue Descriptor

Table 41-8 describes the SSSAR TxQD fields.

Table 41-8. SSSAR TxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	—	Reserved, should be cleared during initialization.
	8	**BNM**	Buffer-not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**UUI**	UUI insertion mode 0 UUI of last CPS packet is 26. 1 UUI of last CPS packet is taken from the next byte after the end of the buffer.[2]
	10	**INF**	Indicates the current state of the frame. 0 The next packet will be the first of a new frame. 1 Currently in the middle of the frame.
	11	**CPS**	Sublayer type. For an SSSAR TxQD, this field must be cleared. 0 SSSAR. 1 CPS packet.
	12	**TBM**	Transmit buffer interrupt mask for TxBD table. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue.) 1 The transmit buffer event of this queue is enabled.
	13	**SSSAR**	SSSAR bit. This bit must be set.
	14–15	—	Reserved, should be cleared during initialization.
0x02	0–7	**Seg_Len**	Specifies the maximum length in bytes of the SSSAR PDU (excluding the packet header). Seg_Len is limited to 45 in NoSTF=1 mode. The QUICC Engine block always attempts to segment the SSSAR SDU according to this length, but if the buffer is ended then the packet length can be less than this value.
	8–15	—	Reserved, should be cleared during initialization.
0x04	—	**TxBD Table Base**	Must be initialized to the first TxBD by the user. Should be 8 byte aligned.
0x08	—	**TxBD Table Offset Out**	Used to calculate the pointer to the next TxBD to be used for transmission. Should be cleared during initialization.
0x0A	—	—	Reserved, should be cleared during initialization.

Table 41-8. SSSAR TxQD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x0C	—	**NextQueue/Next QD_Offset**	Points to the next TxQD to be serviced after this one. See Section 41.3.2, "Transmit Priority Mechanism". In case TCT[ExtTQD]=1 this entry acts as an offset from TxQDExtBase address multiplied by 32. See Section 41.3.5.2, "External TxQD versus Internal TxQD".
0x0E–0x1f	—	**—**	Reserved, should be cleared during initialization.

1 Entries in boldface must be initialized by the user.

2 The 5-bit UUI field is inserted into the header of the last packet of an SSSAR SDU. The user must append the UUI to the last buffer as an additional byte. This additional byte is then inserted into the UUI field of the last packet header (note that, it is important that this additional byte—the byte after the last byte in the last data buffer of the SSSAR frame—contains the 5 bits of the UUI). The 3 MSB bits of this extra byte should be cleared; refer to Figure 41-15.

41.3.5.6 SSSAR Transmit Buffer Descriptor

The SSSAR buffer structure consists of a BD table that points to data buffers. The buffers may contain SSSAR SDUs belonging to different CIDs. Each buffer may contain a whole SSSAR SDU or part of it. The CPS CID is located in the first BD of the SSSAR SDU. See Figure 41-17.

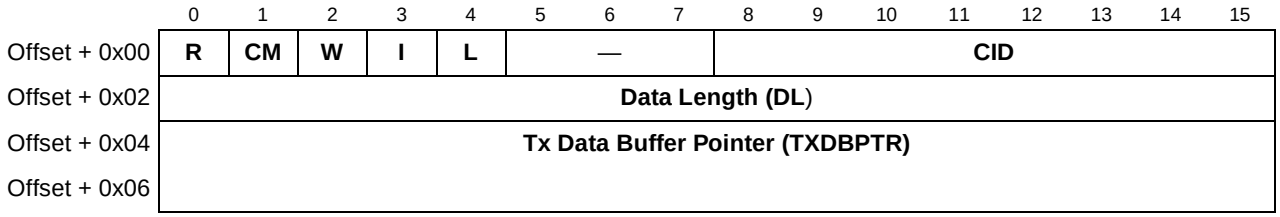


Figure 41-17. SSSAR TxBD

Table 41-9. SSSAR 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	**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 the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the QUICC Engine block transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The UCC[GINTx] bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	**L**	Last. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–7	—	Reserved, should be cleared during initialization.
	8–15	**CID**	Contains the CID number of the SSSAR SDU pointed by this BD. This field is only valid if this is the first BD of a SSSAR SDU.
0x02		**Data Length**	Contains the length of the buffer associated with this BD. If this is the last buffer (L=1) and the UUI bit in the SSSAR TxQD is set, the 5-bit UUI field is located at (TXDBPTR+Data Length)[3:7] with bit [3] being the msb, that is, in the byte (right justified) immediately following the last byte of the buffer. For best bandwidth utilization and optimized partioning of the SDU to packets of exactly SegLen size when SDU is spread over multiple BD's, the application should set Data Length to be an integer multiply of SegLen. (Data Length == n* SegLen). For an SDU on a single BD this recommendation does NOT apply.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the QUICC Engine block.

NOTE: Entries in boldface must be initialized by the user.

41.3.5.7 Transmitter Statistics Structure

The structure of transmitted CPS packet / SSSAR frame counters per CID in each Transmit VC is illustrated in Figure 41-18. This mode is enabled ONLY if TCT[StatsEn] is set. Each Entry in the TxCIDstatsOffset / TxCIDstatsAddr Table gives the location of a set of up to 256 - 16 bits counters

representing a per CID transmitted statistics for CPS packets / SSSAR frames. The first Entry in each Statistic Table (“CID0” Entry) is designated for ATM Cell Counter for this AAL2 VC.

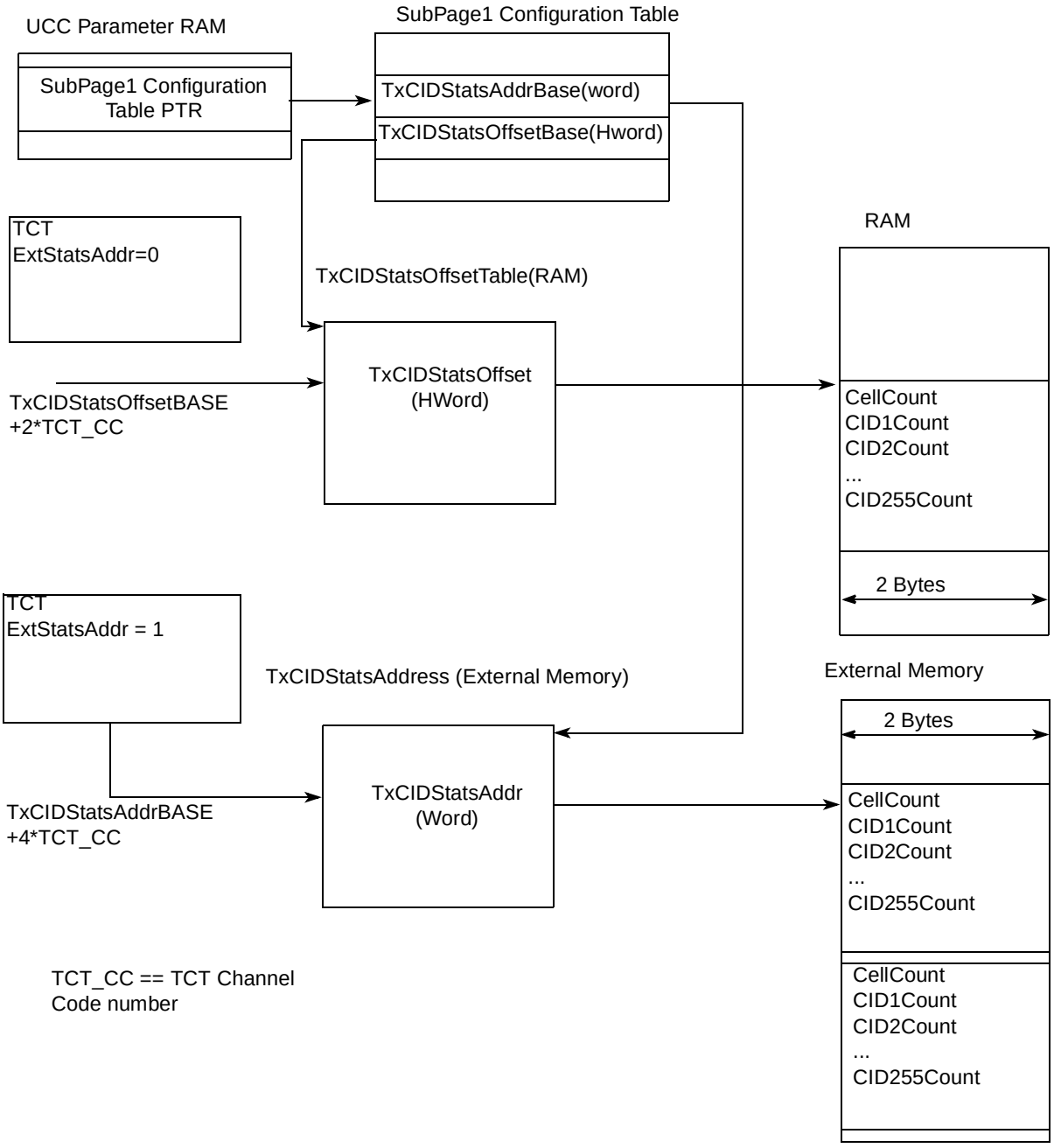


Figure 41-18. TxCIDStats and CellStats structure

As described in Figure 41-18, there are two possible ways to access the TxCID Statistics counters:

1. TCT[ExtStatsAddr] =0 - Using TxCIDStatsOffset, see Table , and TxCID Statistics Tables, which resides in internal multi-user RAM. In this case the access to the TxCIDstatsOffset Entry is by adding TxCIDStatsOffsetBase + 2*TX_CH. The TxCIDStatsOffset Entry is of 2 bytes length, and is used as the direct RAM address of the TxCIDStatistics Table’s entry related to the specific TX

channel (which consist up to 256 CID counters). In order to reach the specific CID counter, use TxCIDStatsOffset + 2*CID. The user could use this method, for a limited number of Tx channels and CID's statistics counters, since this method has better performance than the second one, but consumes a lot of RAM resources.

2. TCT[ExtStatsAddr] =1 - Using TxCIDstatsAddr, see Table , and TxCID Statistics Tables, which resides in external memory. In this case the access to the TxCIDstatsOffset Entry is by adding TxCIDStatsAddressBase + 4*TX_CH. The TxCIDstatsAddr Entry is of 4 bytes length, and is used as the direct External address of the TxCIDStatistics Table's entry related to the specific TX channel (which consist up to 256 CID counters). In order to reach the specific CID counter, use TxCIDStatsAddr + 2*CID. This method can support all 64K Channels in the systems, and doesn't consumes any of the internal multi-user RAM resources, since all the relevant tables resides in external memory. Bus is selected by TCT[BDB].

A Transmitter interrupt is generated when there is an overflow of any of the TxCID Statistics counter or of the Tx ATM cell counter. In this case the TxOF bit is set in the interrupt Entry, and the relevant CID is indicated - for Tx ATM cell overflow the CID field is equal to 0. See Table 41-21 for further information.

NOTE

User can use less than 255 CIDs per TCT channel, since the 2 different mechanisms described above uses a direct memory pointer, to internal or external memory. In this way there should be No 'holes' in memory which are dedicated for unused CIDs.

41.4 AAL2 Receiver

The following sections describe the AAL2 receiver.

41.4.1 Receiver Overview

The receiver cycle starts after the UCC receives a cell. If the cell header is successfully mapped to an ATM channel number, the corresponding RCT is fetched and the AAL type is read. For AAL2 cells, the receiver begins by checking if the last cell received in this channel has an uncompleted (split) CPS packet. If so, the receiver first finishes handling this packet.

The receiver then goes through a validation process. The receiver matches the OSF field in the STF with the expected OSF based on the actual split packet (if the first packet is not split, the OSF should be zero). If the two values do not match, an OSF error interrupt is issued and the receiver drops the last packet. Also, if the STF parity check, or the OSF>47 check results in an error, the receiver issues an interrupt and discards the whole cell. For an SN error the split, or part, of the cell is discarded, an interrupt is issued, and the rest of the cell is being processed as usual.

If any of the above errors has occurred and the cell has started with the remainder of an uncompleted packet, the receiver does the following:

- For a CPS sublayer CID, the packet's RxBD[UP] (uncompleted packet) bit is set.
- For an SSSAR sublayer CID, the buffer is closed with RxBD[RxError = US = 10] (uncompleted SDU), and the rest of the frame is dropped.

The receiver now begins the process of extracting new CPS packets out of the cell with another round of error checking. The receiver examines each CPS packet header for the following errors:

- Incorrect packet HEC. The packet and rest of the cell are discarded.
- Packet length (LI+1) is larger than CPS_Max_SDU_Length. The receiver discards the packet and then continues to extract the next packet in the cell. However, if the packet belongs to an SSSAR CID, the receiver closes the SSSAR buffer with RxBD[RxError = OS = 11] (oversized) and discards the rest of the frame.

The receiver issues an interrupt for each of the above errors.When a SSSAR buffer is closed with RxBD[RxError = US or OS], indicating Uncompleted SDU or Oversized, then RxBD[L] is set, and if RxQD[RFM]=1 then the receiver also issues an RXF interrupt.

Then, if no errors have occurred in the packet header, the packet CID is used to match the PHY | VP | VC | CID with an RxQD; see Section 41.4.2, “Mapping of PHY | VP | VC | CID.” The match process yields an RxQD pointer. The RxQD indicates the type of SAR operation to be performed on the PHY | VP | VC | CID. Three SAR operation modes are supported:

- For the CPS sublayer, each packet from the CID is stored, as is, in a one packet buffer.
- For switching, the packet is stored directly into the transmit buffer, and the CID is translated.
- For the SSSAR sublayer, all packets received from the CID are reassembled into an SSSAR SDU similar to the BD and buffer structures used for AAL5 frames.
- For the SSSAR sublayer, last packet UUI indication is stored in last RxBD of the SDU.

For the SSSAR sublayer, two additional parameters are verified for each new packet received:

- RAS_Timer expiration. The RAS_Timer_Duration defined in the AAL2 parameter RAM limits the time (starting when the first packet is received) allowed to receive a complete SSSAR SDU. If this time limit is exceeded, the receiver closes the current buffer with RxBD[RxError = TE = 01] (Ras_Timer expired) and starts a new SSSAR frame with the next packet. When a buffer is closed with RxBD[RxError = TE = 01], RxBD[L] is not set and the receiver does not issue an RXF interrupt.
- SSSAR_Max_SDU_Length. With each new packet the receiver checks whether the current accumulated length of the SSSAR SDU exceeds the SSSAR_Max_SDU_Length. If so, the receiver closes the current buffer with RxBD[RxError = OS = 11] (oversized), discards the rest of the SSSAR SDU, RxBD[L] is set, and if RxQD[RFM]=1 issues an RXF interrupt.

NOTE

The user must not share the same RxQD table for multiple CIDs from different Rx ATM Channel or for multiple CIDs of the same Rx ATM Channel.

41.4.2 Mapping of PHY | VP | VC | CID

The AAL2 mapping mechanism translates a PHY | VP | VC | CID combination into a unique RxQD. There is no support for mapping different combinations of PHY | VP | VC | CID into a single RxQD.

The mapping mechanism, shown in Figure 41-19, can be broken down as follows:

- Each ATM channel number (RCT) has its own CID mapping table. The mapping table can be placed in internal or external memory (according to RCT[MAP]) and is pointed to by RCT[CID Mapping Table Base]. The CID of the received packet is used as an index into the mapping table.
- Each entry in the mapping table contains a 2-byte RxQD offset. This offset, multiplied by 4, is the offset to an RxQD in either the internal or external RxQD table.
- The two RxQD tables serve all the ATM channel numbers of an UCC. (RxQD_Base_Int and RxQD_Base_Ext are defined in the UCC SubPage0 Configuration Table.)
- RxQD offsets from 1 through 511 point into the internal RxQD table located in multi-user RAM at RxQD_Base_Int. Note that the user should not allocate RxQD_Base_Int memory, in case that there are only external RxQDs in use under this UCC. In this case, the external RxQDs are DMA'd into the memory allocated by the RxQD_tmp parameter in UCC/Distributor/Thread Local Parameter Table.
- RxQD offsets greater than 511 point into the external RxQD table located at RxQD_Base_Ext + (512*4).
- Because the three types of RxQDs are different sizes, some offset numbers may not be used.
- RxQD offsets for unused CIDs would be cleared. This way they will be ignored.
- CID MASK mechanism. See Section 41.4.4.4, "CID MASK".

NOTE

RxQDs should be located in 4 byte aligned addresses. But it is recommended that external RxQDs will be located in addresses that are aligned to their size, inorder to optimize the bus latency,

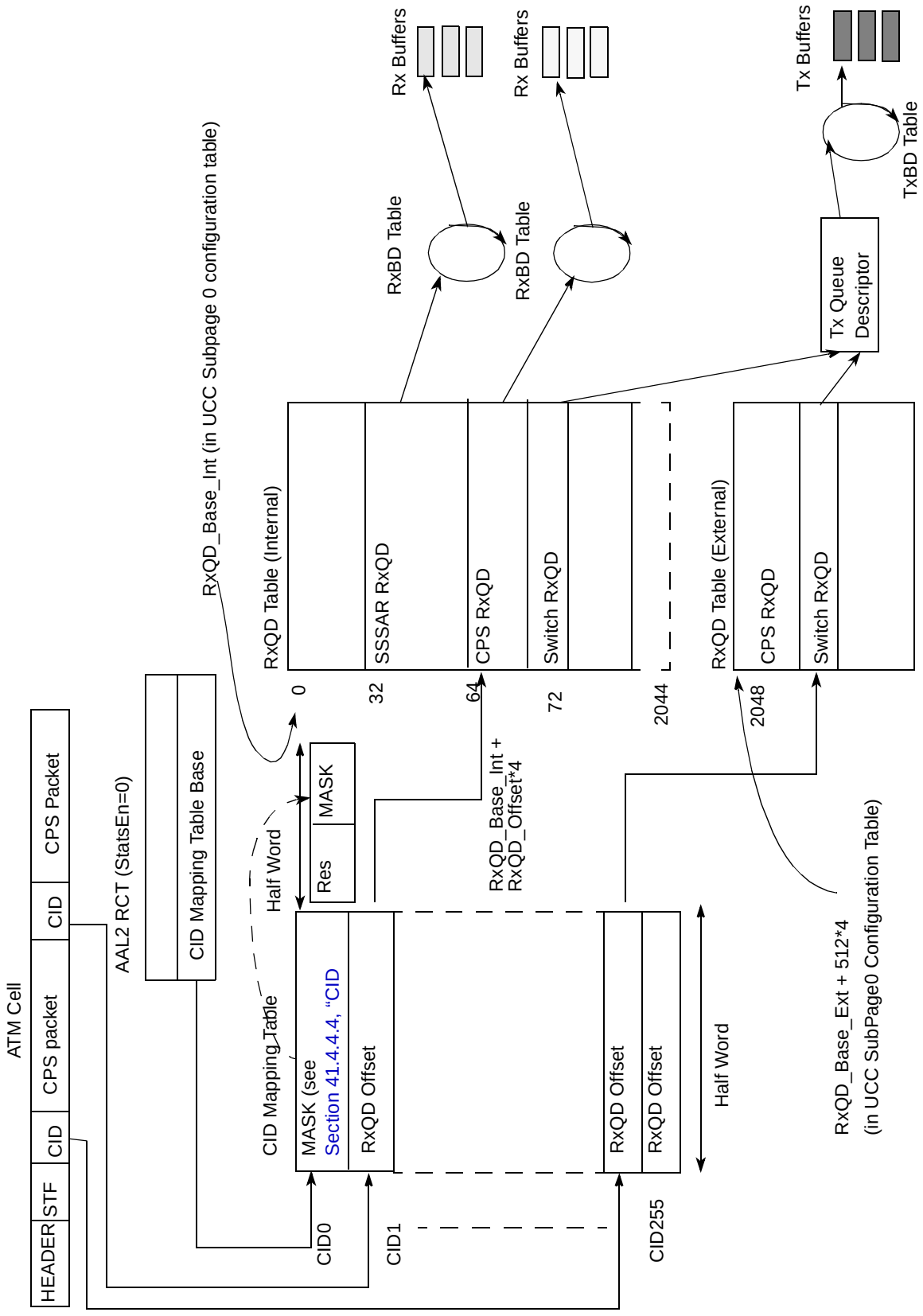


Figure 41-19. CID Mapping Process

41.4.3 AAL2 Switching

Two types of descriptors are available. Both descriptors hold a translation CID number. Type-1 descriptor has a pointer to a CPS TxQD which resides in multi-user RAM. Type-2 descriptor holds a pointer to a CPS TxQD which resides in external memory.

The TxQD structure for both types holds the BD ring management which is common both to the receiver and transmitter. Offset_in is handled by the receiver and offset_out by the transmitter. The transmit scheduling of the packet is done by the APC according to the programmed bit rate of the ATM channel that holds the switched queue.

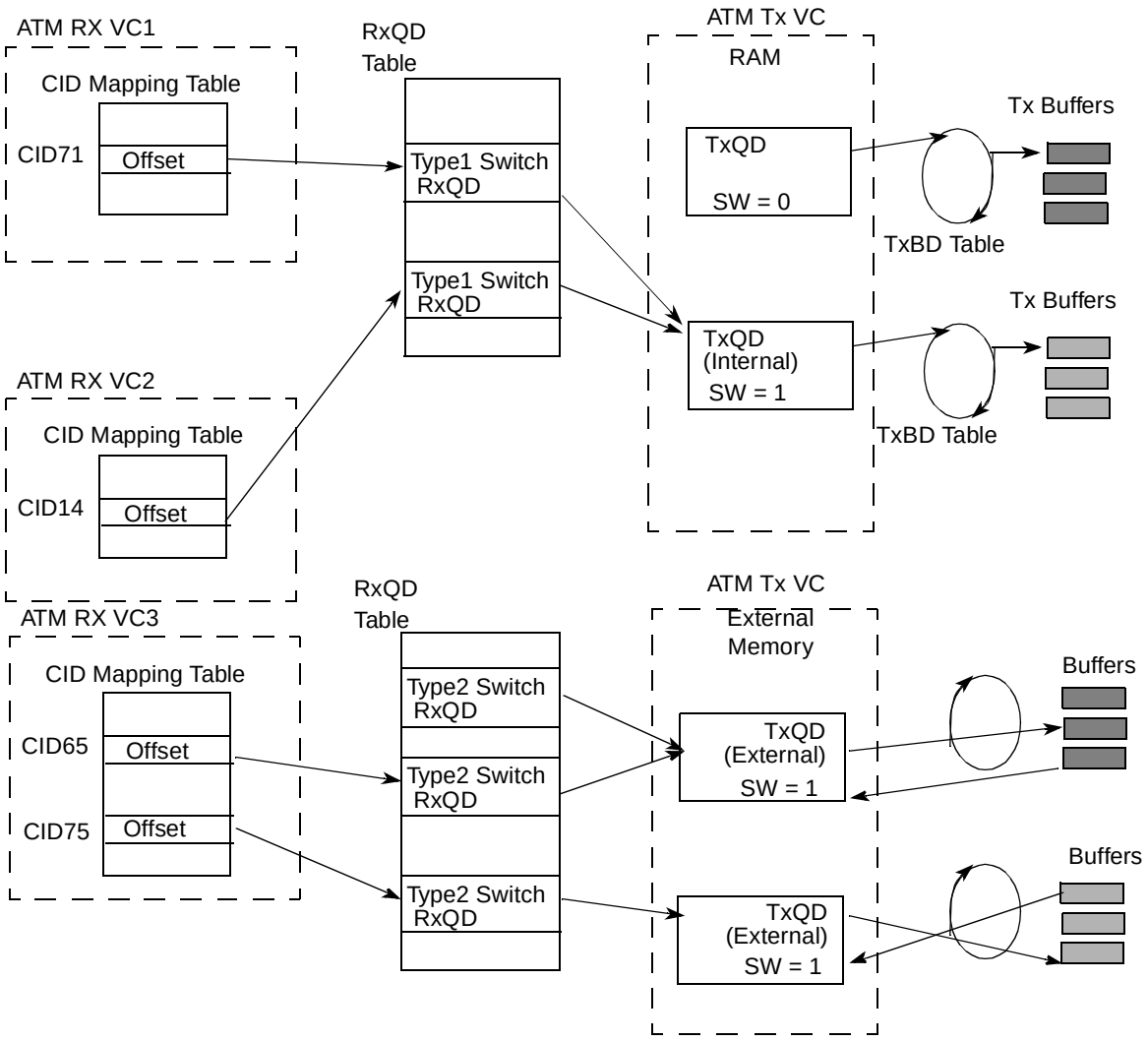


Figure 41-20. AAL2 Switching

A mechanism of Automatic activation and deactivation of ATM channels (Auto VC ON and Auto VC OFF) is defined for the AAL2 switching. In this mode when the receiver receives a packet, the corresponding TCT channel, which is the channel that is pointed by the RxQD[TCC] field, is activated (assuming that this channel was not already in the APC scheduling table) by inserting the TCT channel

code into the APC scheduling table. When the ATM transmitter has a situation where there is no packets ready for transmission in the queue (TxQD) the channel is automatically removed from scheduling table. The RQD data structure for this mode is different then the standard switched RQD. This mechanism is enabled, by setting RxQD[AVCON] bit and configuring the TAPTR and TCC parameters in the RxQD on the receiver's side, and setting TCT[AVCF] bit and clearing TCT[VCON] bit on the transmitter side.

NOTE

AVCON/AVCF mechanism does not supported for GBR mode.

When working in traditional APC AVCON/AVCF mode, an internal data structure- AVCON table should be allocated and cleared, see Table , for the ATM_AVCON_BASE description. The AVCON table contains one bit per TCT channel. For better memory utilization it is recommended that these TCTs are consecutive, to avoid gaps in the AVCON table. The area dedicated for this table is equal to the integer "round up" number of switched TCTs (which participate in the AVCON/AVCF mechanism) divided by 8. The transmitter clears the corresponding TCT bit on event of deactivating a channel, and the receiver, upon detecting that this bit is cleared, will activate the channel and set this bit.

IMPORTANT - When working in GCRA scheduler AVCON/AVCF mode, the ATM_AVCON_BASE parameter and table allocation is **not** necessary, since we already have the priority MASK bits, which have the same meaning as the ATM_AVCON_BASE parameter.

NOTE

When working in switch AVCON/AVCF mechanism:

1. Under the TCT there should be only TxQDs of switch mode.

2. It is possible in some scenarios that a packet would be stuck in the BD ring till the next packet would be received to the same queue - this can introduce a transmission packet delay.

A partial packet discard mode is provided for the AAL2 switched channels that perform end-to-end SSSAR. When this mode is enabled (switch RxQD[PPD]=1), if no buffer is available to receive a packet in the middle of a frame, the subsequent middle packets of the SSSAR SDU are discarded. When the last packet of the SSSAR SDU arrives, the receiver attempts to re-open a buffer.

The receiver will mark an open BD which is partially filled as UP (Un-completed packet). In normal operation when the packet is fully received, the UP mark is removed, the Empty bit is set and switching operation continues. In case of an error which is detected by the receiver the BD is marked as Empty (set to 1) and the UP indication is still on. In this case the transmitter will skip this BD and proceed to the next one.

If the transmitter is currently pointing to a BD that is marked as UP (part of the packet is received and reception of the rest of the packet has not yet occurred), the transmitter cannot advance past this point until the remaining data of the packet has arrived or there is an error detected in the receiver. See Figure 41-30.

In cases where there is aggregation of several receiver channels into a single transmitter TxQD, a common BD ring is shared. A slow receiver may create stall conditions on the switched transmitter due to partial packet conditions. That is, a slow channel receives a partial packet and will mark the BD as UP. Faster channels may fill the BD ring before the remainder of the partial packet from the slow channel is

completed thus data will be dropped from the faster channels, creating RxBSY conditions. One way to compensate for the stalled conditions would be to increase the BD ring size.

If a channel stops receiving data in the middle of a packet, the transmitter will be in a stuck state as it cannot advance past the BD with UP set until the rest of the packet has been completed or a receiver error is detected. This would require host intervention to unstick the stuck BD ring.

In order to avoid the transmitter from getting stuck state and to minimize stalling, one can make use of the TBNR Time-out mechanism, which is enabled in the RCT[TBNR Time-Out CNT] field. The receiver marks the BD with the Time-Out value upon entering a partial packet state on a BD. The transmitter monitors this counter and decrements it on each try. When the counter expires the transmitter will skip this "stuck" BD, so the BD ring is available again for the other receivers. See Table 41-11 for more information. The transmitter will also reset the proper receiver from it's "stuck" state so in case this receiver starts operation it will use a new BD from the BD ring.

Another option is to use is Partial Packet Receive Storage mode (PPRS). When this mode is enabled, when a partial packet is received it is written to external memory to offset 'PPRS_EXT_BASE + 64*RCT Channel Code,' see Table . Only when the packet is completely received does the packet get written into the BD ring. This eliminates the UP condition in the BD ring, thus removing the stalled/stuck condition. Using this mode may also minimize the BD ring size as now a slow receiver will not influence the switched transmitter.

41.4.3.1 AAL2 Switch - WFQ PRIORITY Mode

When working in AAL2 Switch mode and WFQ mode is enabled, there are two additional features which are supported:

1. Queue Management.
2. Automatic bandwidth allocation between High and Low priority traffic types. (Real-Time Vs. Best Effort traffic).

NOTE

In systems where multiple CID's are received on different ATM channels are routed to a common TxQD for transmission, and WFQ mode is enabled, the user must enable the PPRS mode.

41.4.3.1.1 Queue Management

There are up to 256 available thresholds sets in the system, each one consists of 8 bytes, that are available for each TxQD. These thresholds sets are placed from 'Thresholds_Base' parameter in Subpage1

Configuration Table, see Table . Each threshold set contains 4 levels (see Figure 41-6). The structure of each threshold set table is described in Figure 41-21 and Table 41-10.

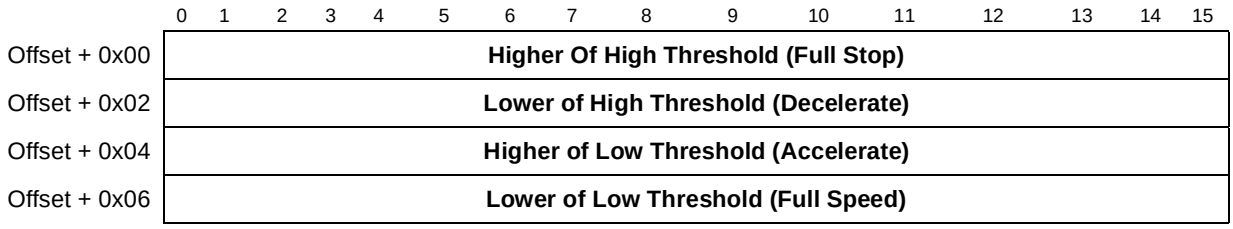


Figure 41-21. Threshold Set Table

Table 41-10. Threshold Set Table

Offset	Name	Description
0x00	**Higher Of High Threshold (Full Stop)**	Higher of High Threshold. The user set the number of packets in queue that corresponds to the Full Stop scenario (queue is too full).
0x02	**Lower of High Threshold (Decelerate)**	Lower of High Threshold. The user set the number of packets in queue that corresponds to the Decelerate scenario (queue is about to be too full).
0x04	**Higher of Low Threshold (Accelerate)**	Higher of Low Threshold. The user set the number of packets in queue that corresponds to the accelerate scenario (queue is about to be too empty).
0x06	**Lower of Low Threshold (Full Speed)**	Lower of Low Threshold. The user set the number of packets in queue that corresponds to the Full Speed scenario (queue is too empty).

There are six types of events each corresponds to a different scenario of threshold crossing, see Figure 41-22. In the following list when one of the thresholds is hit, only the next active threshold that corresponds to it is active, and all the other thresholds are inactive.

1. Full Speed was hit previously, and currently the Decelerate threshold is crossed (crossed means equal or bigger than Decelerate threshold) - Full Speed to Decelerate interrupt is driven.
2. Decelerate was hit previously, and currently the Full Stop threshold is crossed (crossed means equal or bigger than Full Stop threshold) - Decelerate to Full Stop interrupt is driven.
3. Accelerate was hit previously, and currently the Decelerate threshold is crossed (crossed means equal or bigger than Decelerate threshold) - Accelerate to Decelerate interrupt is driven.
4. Full stop was hit previously, and currently the Accelerate threshold is crossed (crossed means equal or less than Accelerate threshold) - Full stop to Accelerate interrupt is driven.
5. Accelerate was hit previously, and currently the Full Speed threshold is crossed (crossed means equal or less than Full Speed threshold) - Accelerate to Full Speed interrupt is driven.
6. Decelerate was hit previously, and currently the Accelerate threshold is crossed (crossed means equal or less than Accelerate threshold) - Decelerate to Accelerate interrupt is driven.

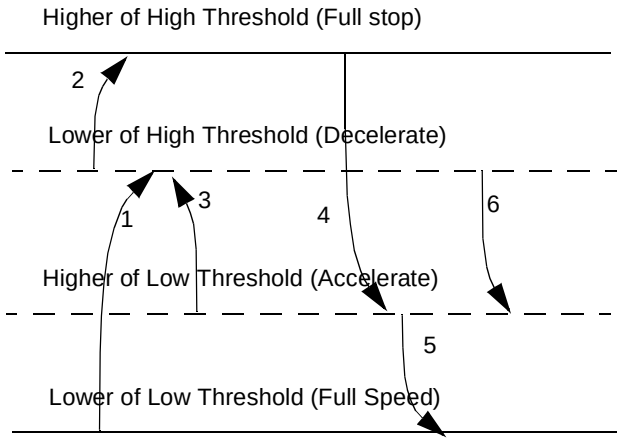


Figure 41-22. Thresholds Events diagram

Each of these events can trigger a maskable interrupt, according to the TxQD[Queue_management] field. See more description of the queue management events in Table 41-22.

The QUICC Engine block implements a hysteresis mechanism, which assures that multiple events are avoided.

NOTE

The threshold set used by each TxQD can be changed dynamically, by writing to the byte which contains the TxQD[Thr_Set_ID] bits.

If the user would like to know which is the specific CID that over-loaded the TxQD, check the Rx CID statistics in the RCT Mapping table of this channel. The user even may have an additional copy of the RCT mapping table, with all the statistics entries cleared, and on event of high threshold interrupt it is possible to switch between these two RCT mapping table bases, by writing to the RCT[CID Mapping Table Base /CIDstatsAddress].

41.4.3.1.2 Automatic Bandwidth Allocation

Up to 4 TxQDs under each TCT can support various types of traffics, as Real-Time (RT) or Best-Effort (BE) traffic. Each queue has a corresponding bit in the ‘RT_MASK’ parameter (Table 41-1), that identifies it as RT (bit set to 1) queue or BE queue. Assuming all of the AAL2 switched channels that use the WFQ mode, have certain guaranteed transmission rate that should not be exceeded by the RT traffic:

The application may require that BE cells not interfere with the RT traffic of other channels in cases of overload. This is achieved by not transmitting the BE channel if the following conditions applies for the channel:

1. RT queues are empty and BE queue non-empty,
2. [Rptr-Sptr]mod slot number > predefined overload limit, and
3. The overload limit parameter in the WFQ Table (see Table 41-1) is not zero

These conditions implies that the system is overloaded, and the objective is to decrease the BE traffic rate by not transmitting the BE cells. The channel is rescheduled for transmission based on PCR without transmitting any data at the current time.

Figure 41-23 depicted the operation of the APC.

NOTE

The same mechanism applies for GCRA scheduler mode, but instead of checking RPTR-SPTR in time slots, the TSR-[Channel FT] is checked in TSR units.

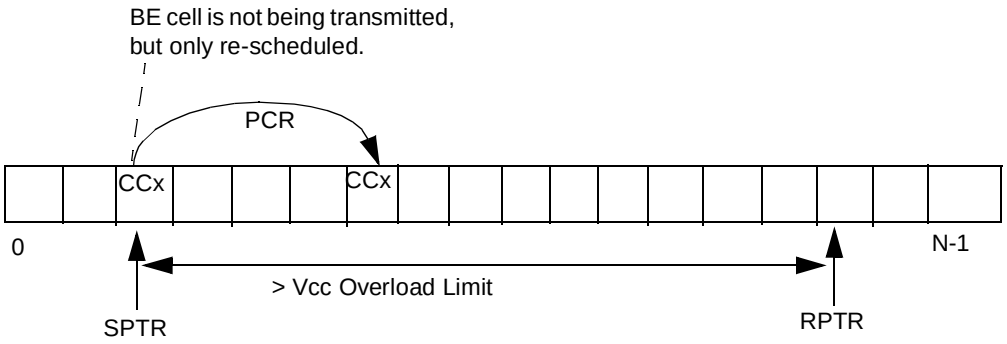


Figure 41-23. Automatic Stop of BE Transmission.

41.4.4 AAL2 RX Data Structures

The following sections describe the RCT and the structures in which CPS packets and SSSAR SDUs are stored in memory.

41.4.4.1 AAL2 Receive Connection Tables

The receive connection table (RCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the RX Queues of the ATM channel are maintained here. The RCT also contains the pointer to the CID mapping table for the ATM VC. Figure 41-24 shows the AAL2-specific RCT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—	—	**GBL**	**BO**		**CETM**	**DTB**	BDB	—	—	**SEGF**	**ENDF**	—	**MAP**	**INTQ**	
Offset + 0x02	—			—		—		—	—	—	—	—	**NoSTF**	**AAL**		
Offset + 0x04	—															
Offset + 0x06																
Offset + 0x08	—															
Offset + 0x0A																
Offset + 0x0c	—															
Offset + 0x0e																
Offset + 0x10	—															
Offset + 0x12	—	—	—	—					—							
Offset + 0x14	—		—						—							
Offset + 0x16	—															
Offset + 0x18	**CID Mapping Table Base /CIDStatsAddress**															
Offset + 0x1A																
Offset + 0x1C	—		**PMT**						**TBNR Time-Out CNT**							
Offset + 0x1E	**Max_CPS_SDU_Deliver_length**								**PPRS**	**WFQ_En**	—	**StatsEn**	**CIDMASK**	**StatsIE**	**EM**	**PM**

Figure 41-24. AAL2 Receive Connection Table (RCT)

NOTE

For an active channel, the QUICC Engine block uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes. Therefore the user can change dynamically the last 8 bytes of the RCT.

Table 41-24 describes the AAL2 RCT fields.

Table 41-11. AAL2 RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, "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 Figure 19-1 and Section 5.4.3.7, "Debug Configuration."
	6	**DTB**	Data buffer and UDC table bus selection. 0 Reside on the coherent system bus. 1 Reside on the secondary bus.
	7	BDB	Bus selection for the BDs, interrupt queues, CID mapping and Statistics table, RxQDs, and the free buffer pool. 0 Reside on the coherent system bus. 1 Reside on the secondary bus. If Type-2 switch mode is being used under this RCT, it implies that this bit value should match the value of the corresponding TCT.
	8–9	—	Reserved, should be cleared during initialization.
	10	**SEGF**	OAM F5 segment filtering 0 Do not send cells with PTI=100 to the raw cell queue. 1 Send cells with PTI=100 to the raw cell queue.
	11	**ENDF**	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12	—	Reserved should be cleared
	13	**MAP**	CID mapping table memory location select 0 Resides in the multi-user RAM. 1 Resides in external memory.
	14–15	**INTQ**	Assigns one of the four available interrupt queues to this ATM channel number.

Table 41-11. AAL2 RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	**NoSTF**	No STF byte. 0 Normal AAL2 cell structure. 1 The cell does not include the STF byte. In this mode each cell starts with a packet and contains only whole packets (no split or part).
	13–15	**AAL**	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol. (Use this configuration for rev A parts only, unless needed for backwards compatibility to existing software development.) 010 AAL5 —ATM adaptation layer 5 protocol 011 AAL3/4 —ATM adaptation layer 3/4 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1 with CES —ATM adaptation layer 1 protocol with optional support for circuit emulation service. (For rev B and later parts, this configuration is strongly recommended for all AAL1 applications, with or without CES.) All others reserved.
0x04 - 0x17	—	—	Reserved, should be cleared during initialization.
0x18	—	**CID Mapping Table Base /CIDstatsAddress**	For StatEn=0: Points to the base address of the CID mapping table (see Figure 41-19). If RCT[MAP] = 0, the pointer contains a multi-user RAM address and the MSB Hword (bytes in addresses 0x18 and 0x19) **must** be cleared. If MAP = 1, the pointer is a full 32-bit address in the memory space. For StatEn=1: Points to the location of the CID mapping and statistics table, which contains 256 (or less) received CID statistics counters and CID RxQD Offsets, per VC. See Figure 41-25. RCT[MAP] bit still has the same functionality as described above.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PMT**	Performance monitoring table. Assigns one of the available 64 performance monitoring tables to this VC. The table's starting address is PMT_BASE+PMT*32.
	8–15	**TBNR Time-Out CNT**	The **TBNR Time-Out CNT** is a parameter that describes the amount of attempts the transmitter tries to transmit a packet on a Switch BD ring which is currently marked as partially filled i.e. waiting for a receiver to finish reception of a packet. This value will be used internally by the transmitter which is the destination for this packet and decremented by the QUICC Engine block on each attempt. Upon reaching the value 1the transmitter will act as if the receiver is stuck in error condition and proceed to the next BD in the BD ring. This parameter is valid in switch mode ONLY and should be programmed to a higher value than the ratio between the transmitter rate and the lowest receiver rate in the BD ring. The 8 bit value is scaled by 4 (setting **TBNR Time-Out CNT=1** yields a value of 4) so that the max number is 1K. Clearing this field will disable this feature completely. **Note:** This parameter should be cleared if **PPRS** mode is enabled.

Table 41-11. AAL2 RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1E	0–7	**Max_CPS_SDU_Deliver_Length**	Indicates the maximum size CPS_SDU in bytes that is allowed to be transported on this channel. This value is compared to the length of each CPS_SDU before it is received, as specified in the ITU-T recommendation I.363.2. If the packet length exceeds this value it is discarded, and an interrupt with error_code=5 is asserted.
	8	**PPRS**	Partial Packet Receive Storage. Valid only for switch mode. In this mode when a partially packet is received to a switched RxQD, then instead of writing the packet into the TxQD BD ring immediately, it will be temporarily stored in external memory. Only after the complete packet is received, then it will be stored in the TxQD BD ring. 0 - Partial Packet Receive Storage mode is disabled. 1 - Partial Packet Receive Storage mode is enabled. See more description in Section 41.4.3, “AAL2 Switching”. **Note:** If **PPRS** mode is enabled then the user should clear the **TBNR Time-Out CNT** field.
	9	**WFQ_En**	Weighted Fair Queueing mode Enabled. 0- The Switched RQD doesn’t point to a TxQD which operate in WFQ mode. 1- Switched RxQDs under this RCT **must** point to TxQDs which works in WFQ mode. **Note:** In systems where multiple CID’s are received on different ATM channels are routed to a common TxQD for transmission, and WFQ mode is enabled, the user **must** enable the PPRS mode.
	10	—	Reserved, should be cleared during initialization.
	11	**StatsEn**	Statistics mode Enable bit. 0- CID and Cell counters are disabled for this VC. 1- CID and Cell counters are enabled for this VC.
	12	**CIDMASK**	0 - CID MASK mode is disabled 1 - CID MASK Mode is enabled. See Section 41.4.4.4, “CID MASK”.
	13	**StatsIE**	Statistics Interrupt Enable. 0 - Rx CID Statistics / ATM cell / Switch Packet Discard counters overflow interrupt is disabled. 1 - Rx CID Statistics / ATM cell / Switch Packet Discard counters overflow interrupt is enabled. See Section 41.4.4.3, “Receiver Statistic Structure—Expanded CID Table”.
	14	**EM**	Receive error mask for AAL2 protocol-specific events. Note that buffer-not-ready, Rx buffer, Rx frame and Rx Statistics events are not affected by this mask. 0 Disable AAL2 receive error events. 1 Enable AAL2 error events.
	15	**PM**	Enable performance monitoring 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received by this VC, the associated performance monitoring table is updated.

NOTE: Entries in boldface must be initialized by the user.

41.4.4.2 CID Mapping Tables and their RxQDs

Each PHY | VP | VC | CID combination is assigned an RxQD using a CID mapping table. To multiplex several receive CIDs into a single common queue, map each multiplexed PHY | VP | VC | CID combination to one RxQD.

The ATM channel's RCT contains the base address of the associated CID mapping table. This base address is external (32 bits) when RCT[MAP]=1; otherwise, when RCT[MAP]=0 the table resides in the multi-user RAM and the base address is two bytes. The CID of the received packet is used as an index into the mapping table. The mapping table entries are 2-byte RxQD offsets. If the CID mapping table is external, it must be on the same bus as the BDs and interrupt queues as specified by RCT[BDB].

RxQD offsets between 1–511 point into the internal RxQD table located in multi-user RAM at RxQD_Base_Int. The address of an internal RxQD is RxQD_Base_Int (see Table 41-17) + 4*RxQD_Offset. Note that the user should not allocate RxQD_Base_Int memory, in case that there are only external RxQDs in use under this UCC. In this case, the external RxQDs are dma'd into the memory allocated by the RxQD_tmp parameter in UCC/Distributor/Thread Local Parameter Table.

Offsets between 512–65535 belong to the external RxQD table. The address of an external RxQD is RxQD_Base_Ext, see Table 32-18, + 4*RxQD_Offset. The external RxQD table must be on the same bus as the BDs and interrupt queues as specified by RCT[BDB].

RxQD offsets for unused/unwanted CIDs should be cleared. This will ensure that data is not received on these CIDs.

Because the three kinds of RxQDs are each a different size (for example, an SSSAR RxQD is 32 bytes and a CPS switch RxQD could be only 4 bytes), some of the offset numbers are left unused.

NOTE

RxQDs should be located in 4 byte aligned addresses. But it is recommended that external RxQDs will be located in addresses that are aligned to their size, in order to optimize the bus latency,

41.4.4.3 Receiver Statistic Structure—Expanded CID Table

When RCT[StatsEn]=1 the CID mapping table changes it's format and it contains statistical data gathering in addition to the regular CID mapping information, for each CID under this VC. The structure contains counters of received CPS packets / SSSAR frames per CID in each Receive VC. In this case the entry RCT [CID Mapping Table Base] in the RCT acts as a pointer to this data structure. It is described in Figure 41-25. The first Entry in the Statistic Table ("CID0" Entry) is designated for ATM Cell Counter per AAL2 receive VC. Accessing to this table is by using the RCT[CIDstatsAddress] pointer - if RCT[MAP] equals 0 then use only 2 least significant bytes of this address, that points to internal map offset, else if RCT[MAP] equals 1 then use the all 4 bytes entry as an external memory pointer. The process of fetching the RxQD after locating the RxQDOffset remains exactly the same as was defined in Section 41.4.4.2, "CID Mapping Tables and their RxQDs".

The Structure is defined as a max of 256 entries each entry has two fields, a 16 bit CID counter and the RxQD offset which is similar to the CID mapping table.

A Receiver interrupt is generated when there is an overflow in any of the RxCID Statistics counters or the Rx ATM cell counter. In this case the RxOF bit is set in the interrupt Entry, and the relevant CID is indicated - for Rx ATM cell overflow the CID field is equal to 0. See Table 41-21 for further information.

Note - The structure of the Expanded CID mapping table could contain less than 256 entries, since the mechanism of CID MASK, See Section 41.4.4.4, "CID MASK". RxQD offsets for unused CIDs would be

cleared. This way they will be ignored, and their Statistic counters wouldn't be updated.

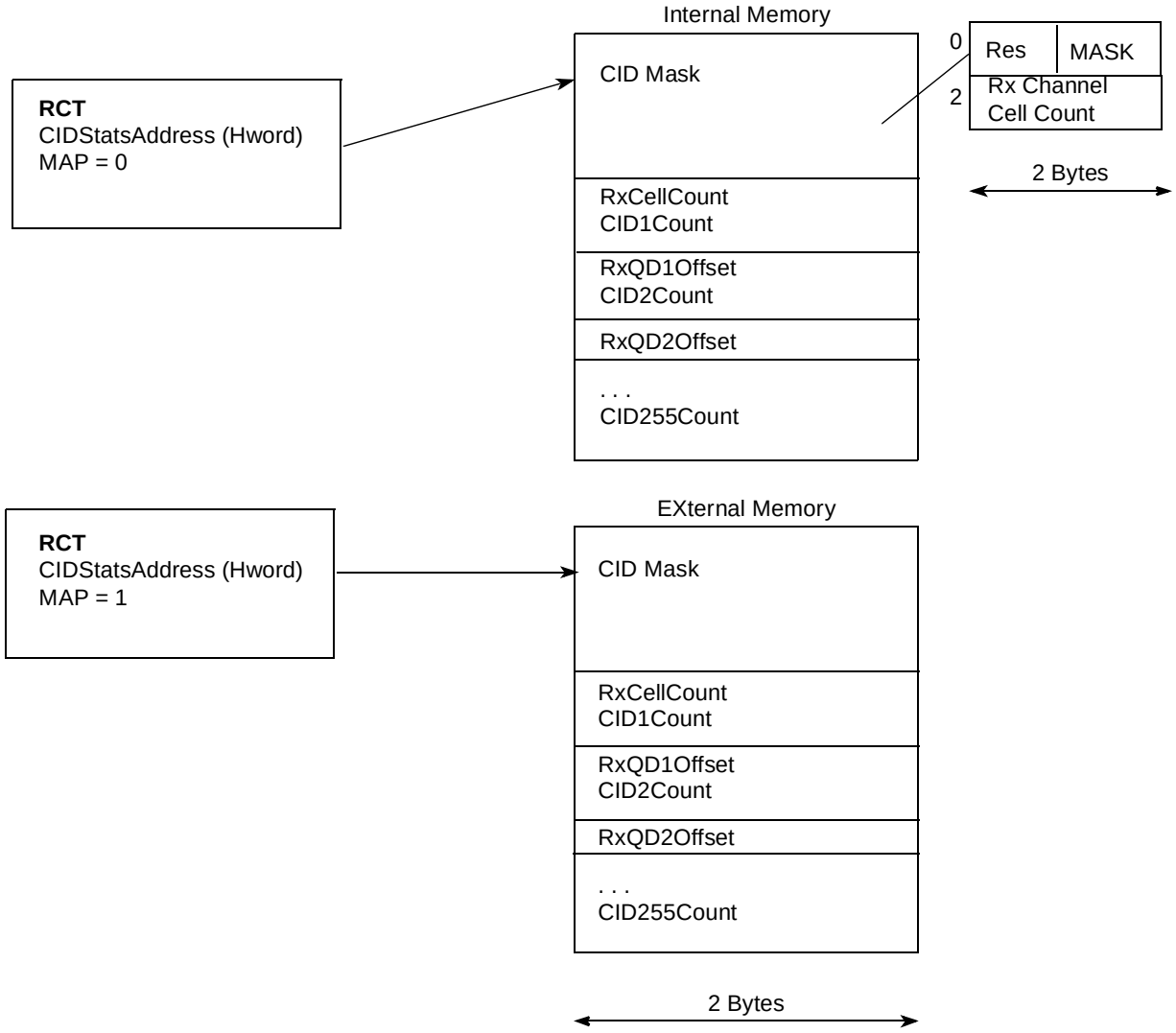


Figure 41-25. RCT Statistics Structure (Valid Only for RCT[StatsEn] = 1)

41.4.4.4 CID MASK

In order to support fast and efficient discard for invalid CIDs, there is a 8 bit CID mask value per receiver channel. Any packet with a CID value outside the mask value is discarded. i.e. writing 0 to a bit in the mask implies that any incoming packet with CID that has this bit value equal to 1 will be discarded. This could decrease the size of the CID Mapping/Statistic Table as there is no need to program all entries in the table. This mode is enabled by setting RCT[CIDMASK].

The MASK is located at the second byte of the first 16 bits entry ("CID0" location) of the CID Mapping table (See Figure 41-19) if RCT[StatsEn]=0, or of the Expanded CID mapping table (See Figure 41-25) if RCT[StatsEn]=1.

Notice - As described in Section 41.4.2, "Mapping of PHY | VP | VC | CID", RxQD offsets for unused CIDs would be cleared. Therefore also the Receiver CID Statistics counters wouldn't be incremented.

41.4.4.5 CPS Rx Queue Descriptors

The CPS RxQDs are grouped into RxQDs Tables (one for internal queues and one for external). Each CPS RxQD, as shown in Figure 41-26, points to an CPS RxBD table.

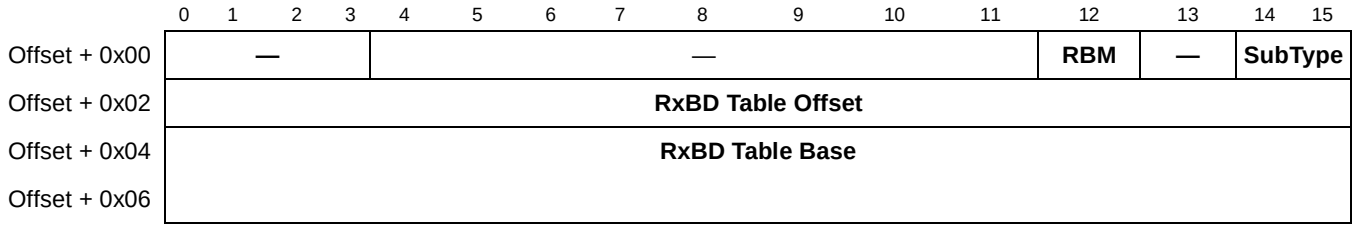


Figure 41-26. CPS Rx Queue Descriptor

Table 41-12 describes the CPS RxQD fields.

Table 41-12. CPS RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–11	—	Reserved
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	13	—	Reserved, should be cleared during initialization
	14–15	**SubType**	SubType. Sublayer type, should be 00 (CPS) for this descriptor. 00 CPS sublayer. 01 CPS switched. 10 SSSAR. 11 Reserved.
0x02	—	**RxBD Table Offset**	Holds the offset to the next BD to be opened by the receiver. Should be cleared during initialization.
0x04	—	**RxBD Table Base**	Holds the pointer to the first BD in the BD table. Should be 8 byte aligned.
NOTE: Entries in boldface must be initialized by the user.			

41.4.4.6 CPS Receive Buffer Descriptor (RxBD)

The CPS buffer structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 41-27. The buffers contain the packet payload.

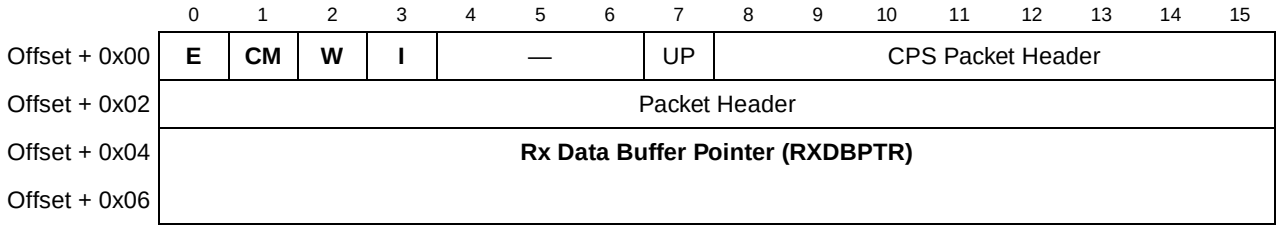


Figure 41-27. CPS Receive Buffer Descriptor

Table 41-13 describes the CPS RxBD fields.

Table 41-13. CPS RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**E**	Buffer empty bit. 0 The CPS RX buffer is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The QUICC Engine block does not use this BD while E remains zero. 1 The CPS RX buffer is empty or reception is in progress. This is controlled by the QUICC Engine block. Once E is set, the core should not access any fields of this buffer.
	1	**CM**	Continuous mode 0 Normal operation. 1 The QUICC Engine block does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the QUICC Engine block next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the QUICC Engine block receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt. 0 The QUICC Engine block will not issue an interrupt after this buffer is serviced. 1 The QUICC Engine block will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4-6	—	Reserved, should be cleared during initialization
	7	UP	Uncompleted packet. 0 No error occurred in this packet. 1 A receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue.
	8-15	CPS Packet Header	Contains the beginning of the packet header. See Figure 41-15 for the CPS packet header format.
0x02		CPS Packet Header	Contains the rest of the packet header. The QUICC Engine block checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 41-15 for the CPS packet header format.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the QUICC Engine block.

NOTE: Entries in boldface must be initialized by the user.

41.4.4.7 CPS Switch Rx Queue Descriptors

Switch RxQD, shown in Figure 41-28 and Figure 41-29, are used for CIDs that are being switched from one $PHY_1 \mid VP_1 \mid VC_1 \mid CID_1$ to another $PHY_2 \mid VP_2 \mid VC_2 \mid CID_2$. The switch RxQD contains the translation CID that is sent with the packet in the transmit buffer. A PPD mode enables the discarding of the rest of an SSSAR frame when a buffer is not available. It is possible to aggregate multiple RQD's

(CID’s residing under the same ATM Rx channel) into a single TxQD. i.e. aggregate multiple CID’s into a single ATM channel.

The switch RxQD contains the translation CID that is saved with the packet in the transmit buffer. A PPD mode enables the discarding of the rest of a SSSAR frame when a buffer is Note that the CPS switch RxQD must be unique for every Rx switched CID.

Two types of RQD’ are available:

Type-1: This RxQD contains a pointer to a TxQD which resides in the RAM. (Internal TxQD’s).

Type-2: This RQD points to an external TxQD. The size of this data structure is expanded now as it points to an address in external memory.

The switch RxQD size could be 4 bytes (Type-1) or 8 bytes (Type-2), in case that the AVCON/AVCF mechanism is disabled, or it could be 8 bytes (Type-1) or 12 bytes (Type-2), in case that the AVCON/AVCF mechanism is enabled (see Section 41.4.3, “AAL2 Switching”).

	0 1 2 3 4 5 6 7	8	9	10	11	12	13	14 15
Offset + 0x00	**TX CID**	**AVCON**	**GCRA_AVCON**	**SWTYPE**	**—**	**RBM**	**PPD**	**SubType**
Offset + 0x02	**TxQD Pointer/WFT#i_ptr**							
Offset + 0x04	**TCC**							
Offset + 0x06	**TAPTR (Traditional APC)**							
	GCRA Scheduler PHY Parameter Table ptr (GCRA_Sched)							**PRI (GCRA_Sched)**

Figure 41-28. Type-1 CPS Switch Rx Queue Descriptor

Table 41-14 describes the CPS switch RxQD fields.

Table 41-14. Type-1 CPS Switch RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0-7	**TX CID**	Translation CID. The received CID is saved in a TX Queue with this new CID number.
	8	**AVCON**	Auto VC On. Automatic Scheduling of a channel into the APC. If set, the receiver automatically schedules, if needed, the TCC into the APC scheduling table. Valid also for GCRA scheduler mode. 0 Automatic scheduling is disabled. 1 Automatic scheduling is enabled. See Section 41.4.3, “AAL2 Switching”.
	9	**GCRA_AVCON**	GCRA Scheduler AVCON. Valid only if AVCON bit is set. Indicates which type of APC is used to activate the channel. 0 - Traditional APC. 1 - GCRA Scheduler.
	10	**SWTYPE**	Switch Type- For this RQD’s this bit should be cleared. 0 Type 1. 1 Type-2
	11		Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	13	**PPD**	Partial packet discard. 0 Normal mode. 1 When a buffer-not-ready event causes a packet to be discarded, the remainder of the SSSAR SDU is also discarded. This allows for better performance for switched channels that implement SSSAR.
	14-15	**SubType**	Sublayer type. Should be 01 (CPS switched) for this descriptor. 00 CPS sublayer. 01 CPS switched. 10 SSSAR. 11 Reserved.
0x02	—	**TxQD Pointer /** **WFT#i_ptr**	**If TCT[WFQ_En] is cleared:** Points to the TxQD which resides in RAM into which the packets of this CID are stored and later sent. **If TCT[WFQ_En] is set:** Weighted Finish Time #i Pointer. Pointer to the corresponding TxQD WFT entry. i.e., if TxQD is #i= 2 (#i=0..3), then **WFT#i_ptr = TCT[WFQ_Table_ptr]+4**.
0x04	—	**TCC**	Transmit channel code. Valid only when the switch AVCON/AVCF mechanism is enabled (see Section 41.4.3, “AAL2 Switching”). Used by the QUICC Engine block to automatically insert a connection into the APC scheduling table.

Table 41-14. Type-1 CPS Switch RxQD Field Descriptions (continued)

Offset	Bits	Name	Description
0x06	—		Valid only when the switch AVCON/AVCF mechanism is enabled (see Section 41.4.3, "AAL2 Switching"). This is used by the QUICC Engine block to automatically insert a connection into the APC scheduling table**.**
	0-15	**TAPTR**	**Traditional APC mode:** Transmit APC pointer. Pointer to the APC priority Table entry in multi-user RAM.
	0-13	**GCRA Scheduler PHY Parameter Table ptr**	**GCRA Scheduler mode:** 14 most significant bits of the pointer to the GCRA Scheduler PHY Parameter Table, see Section 32.3.7.1, "GCRA Scheduler PHY Parameter Table."
	14-15	**PRI**	**GCRA Scheduler mode:** Indicates the priority number of the inserted channel.
NOTE: Entries in boldface must be initialized by the user.			

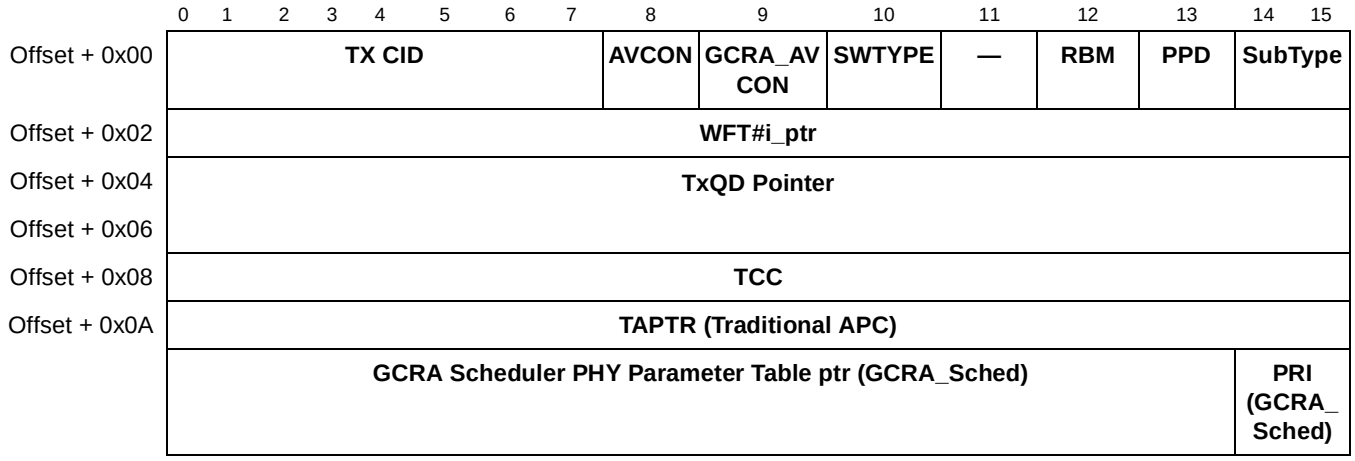


Figure 41-29. Type-2 CPS Switch Rx Queue Descriptor

Table 41-15. Type-2 CPS RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0-7	**TX CID**	Translation CID. The received CID is saved in a TX Queue with this new CID number.
	8	**AVCON**	Auto VC On. Automatic Scheduling of a channel into the APC. If set, the receiver automatically schedules, if needed, the TCC into the APC scheduling table. Valid also for GCRA Scheduler mode. 0 Automatic scheduling is disabled. 1 Automatic scheduling is enabled. See Section 41.4.3, “AAL2 Switching”.
	9	**GCRASched_AVCON**	GCRA Scheduler AVCON. Valid only if AVCON bit is set. Indicates which type of APC is used to activate the channel. 0 - Traditional APC. 1 - GCRA Scheduler.
	10	**SWTYPE**	Switch Type- For this RQD’s this bit should be set 0 Type 1. 1 Type-2.
	11	**-**	Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	14-15	**SubType**	SubType. Sublayer type, should be 01 (CPS switched) for this descriptor. 00 CPS sublayer. 01 CPS switched. 10 SSSAR. 11 Reserved.
0x02	0-15	**WFT#i_ptr**	**Valid only if TCT[WFQ_En] is set:** Weighted Finish Time #i Pointer. Pointer to the corresponding TxQD WFT entry. i.e., if TxQD is #i= 2 (#i=0..3), then **WFT#i_ptr = TCT[WFQ_Table_ptr]+4**.
0x04	—	**TxQD Pointer**	Points to the TxQD which resides in external memory into which the packets of this CID are stored and later sent.
0x08	—	**TCC**	Transmit channel code. Valid only when the switch AVCON/AVCF mechanism is enabled (see Section 41.4.3, “AAL2 Switching”). Used by the QUICC Engine block to automatically insert a connection into the APC scheduling table.

Table 41-15. Type-2 CPS RxQD Field Descriptions (continued)

Offset	Bits	Name	Description
0x0A	—		Valid only when the switch AVCON/AVCF mechanism is enabled (see Section 41.4.3, "AAL2 Switching"). This is used by the QUICC Engine block to automatically insert a connection into the APC scheduling table.
	0-15	**TAPTR**	**Traditional APC mode:** Transmit APC pointer. Pointer to the APC priority Table entry in multi-user RAM.
		GCRA Scheduler PHY Parameter Table ptr	**GCRA Scheduler mode:** 14 most significant bits of the pointer to the GCRA Scheduler PHY Parameter Table, see Section 32.3.7.1, "GCRA Scheduler PHY Parameter Table."
	14-15	**PRI**	**GCRA Scheduler mode:** Indicates the priority number of the inserted channel.
NOTE: Entries in boldface must be initialized by the user.			

41.4.4.8 SWITCH Receive/Transmit Buffer Descriptor (RxBD)

The Switch buffer structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 41-30. The buffers contain the packet payload. This BD is common to the receiver and the transmitter.

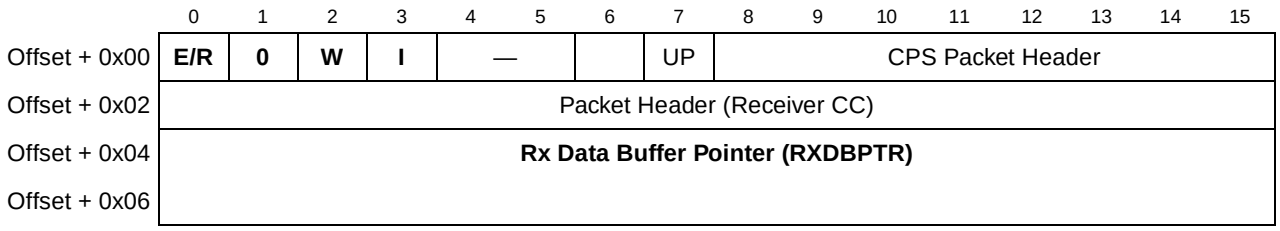


Figure 41-30. Switch Receive/Transmit Buffer Descriptor

Table 41-16 describes the Switch RxBD fields.

Table 41-16. SWITCH RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**E/R**	Buffer Ready bit. 1The RX buffer is full, or data reception was aborted due to an error. The transmitter will transmit the data if the UP bit is cleared i.e. no errors detected. 0 The RX buffer has been transmitted and ready for receiver reception. If the UP bit is set the receiver started reception of part of a packet.
	1	**0**	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 of this current channel. After this buffer has been used, the QUICC Engine block receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt. 0 The QUICC Engine block will not issue an interrupt after this buffer is serviced. 1 The QUICC Engine block will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4-6	—	Reserved. Should be cleared during initialization
	7	UP	Uncompleted packet. 0 No error occurred in this packet and the complete packet has been received. 1 if R/E=1 a receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue. The transmitter will skip this BD when in this state and continue to the next BD in the ring. if R/E=0 a receiver has received the first part of a packet and is waiting for the rest of it to be received on the next ATM cell.
	8-15	CPS Packet Header	Contains the beginning of the packet header. See Figure 41-15 for the CPS packet header format.
0x02		CPS Packet Header (Receiver CC)	Contains the rest of the packet header. The QUICC Engine block checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 41-15 for the CPS packet header format. In case of a “stuck” receiver in **switch** mode, where the BD ring in common to Tx and Rx, this field indicates the last Receiver Channel Code number which has been received. The terminology for “stuck” implies a receiver which started receiving a packet and the rest of the packet hasn’t been received. If the Time-out mechanism is being used this field is being used internally by the QUICC Engine block.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the QUICC Engine block.

NOTE: Entries in boldface must be initialized by the user.

41.4.4.9 SSSAR Rx Queue Descriptor

The SSSAR RxQD, as shown in Figure 41-31, points to the RxBD table and contains other parameters specific to the SSSAR sublayer. This descriptor can belong to only one PHY | VP | VC | CID.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—	—	—	—	—	—		—		—	—	**RasT**	**RBM**	**RFM**	**SubType**	
Offset + 0x02	**RxBD Table Offset**															
Offset + 0x04	**RxBD Table Base**															
Offset + 0x06																
Offset + 0x08	—															
Offset + 0x0A																
Offset + 0x0c	Time Stamp															
Offset + 0x0e																
Offset + 0x10	—															
Offset + 0x12	—															
Offset + 0x14	**MRBLR**															
Offset + 0x16	**Max_SSSAR_SDU_Length**															
Offset + 0x18	—								—							
Offset + 0x1A	—															
Offset + 0x1C	—	—							—	—						
Offset + 0x1E	—															

Figure 41-31. SSSAR Rx Queue Descriptor

Table 41-17 describes the SSSAR RxQD fields.

Table 41-17. SSSAR RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–10	—	Reserved, should be cleared during initialization
	11	**RasT**	Ras Timer enable. 0 Ras Timer disabled (Time Stamp field is still valid). 1 Ras Timer enabled. The Ras Timer duration is set by the Ras Timer Duration parameter in the parameter RAM. If the current SSSAR SDU is not completed before the RasTimer expires, the BD is closed showing the Ras_Timer expired (TE) (SSSAR RxBD[RxError] = 01) and the next packet starts a new SDU.
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	13	**RFM**	Receive frame mask. 0 Disable receive frame interrupt. 1 Enable receive frame interrupt.
	14-15	**SubType**	Sublayer type. Should be 10 (SSSAR) for this descriptor. 00 CPS sublayer. 01 CPS switched. 10 SSSAR. 11 Reserved.

Table 41-17. SSSAR RxQD Field Descriptions

Offset	Bits	Name	Description
0x02	—	**RxBD Table Offset**	Points to the next BD to be handled by the QUICC Engine block. The user should initialize this pointer to zero.
0x04	—	**RxBD Table Base**	Points to the beginning of the BD table. Should be 8 byte aligned.
0x08	—	—	Reserved, should be cleared during initialization.
0x0c		Time Stamp	Used for reassembly time-out of the SSSAR SDU. Whenever the first packet of an SSSAR SDU arrives the timestamp timer is sampled and stored here (regardless of the RasT bit).
0x14	—	**MRBLR**	Maximum receive buffer length. Holds the maximum receive buffer length. The actual buffer size can be less.
0x16	—	**Max_SSSAR_SDU_ Length**	Holds the maximum SSSAR SDU length. Upon each new packet the accumulated frame size is compared with this value. If the limit is exceeded, the QUICC Engine block discards the rest of the packets of the current frame.
0x18-0x1f	—	**—**	Reserved, should be cleared during initialization.

NOTE: Entries in boldface must be initialized by the user.

41.4.4.10 SSSAR Receive Buffer Descriptor

The SSSAR SDU is stored in a BD-buffer structure similar to the structures used for AAL5 frames. The buffer size is determined by SSSAR RxQD[MRBLR]; the actual buffer space used may be smaller. If the received SSSAR SDU is greater than MRBLR, the SDU spans over multiple buffers. UUI of Last packet of the SDU is stored in the RxBD where the L bit set.

The SSSAR RxBD is shown in Figure 41-32.

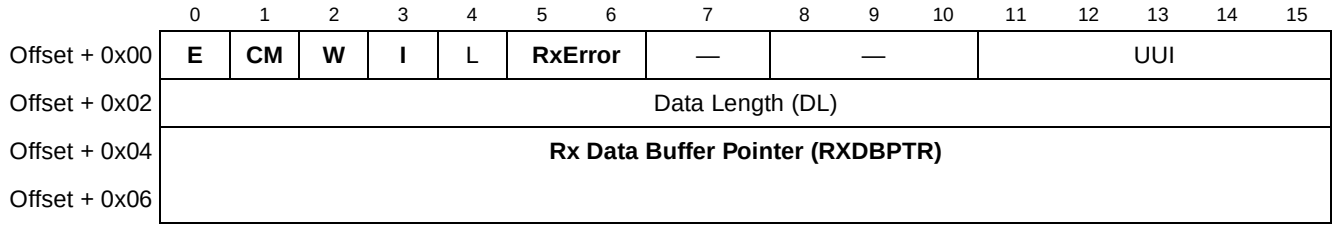


Figure 41-32. SSSAR Receive Buffer Descriptor

Table 41-18 describes the SSSAR RxBD fields.

Table 41-18. SSSAR 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	**CM**	Continuous mode 0 Normal operation. 1 The QUICC Engine block does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the QUICC Engine block next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in the table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the QUICC Engine block receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt. 0 No interrupt is generated after this buffer has been serviced. 1 An Rx buffer event is sent (provided that RxQD[RBM] is set) to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	L	Last. Set by the QUICC Engine block. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5-6	RxError	Rx error occurred 00 No Rx error occurred 01 TE—Ras_Timer expired. The Ras Timer expired before this buffer could be completed. The SSSAR SDU stored in this buffer is not completed.[1] 10 US—Uncompleted SDU. A receive error caused a packet belonging to this SSSAR SDU to be lost. The receiver discarded the rest of this SSSAR SDU. 11 OS—Oversized. The size of the SSSAR SDU has exceeded the Max_SSSAR_SDU_Size parameter. The rest of the SDU was discarded.
	7-10	—	Reserved, should be cleared during initialization.
	11-15	UUI	Contains the UUI of the last packet in received SDU. Valid only where the L bit is set.

Table 41-18. SSSAR RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02		Data Length	Contains the length of the buffer associated with this BD. If this is the last buffer (L=1) of the SSSAR SDU, this field contains the total frame length.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the QUICC Engine block.

[1] When RAS timer expires the RxBD is closed with RAS timer expired indication, the Last (L) bit is not set, and RXF interrupt is not issued. A new RxBD is opened for the next incoming AAL2 packet and the frame is processed as normal and is treated as a new frame. When the next SSSAR end-of-frame indication is received, the RxBD at that time is closed with an L indication, and if RxQD[RFM] = 1, the receiver issues an RXF interrupt.

41.5 AAL2 Parameter RAM

See more description in Table 32-18..

41.6 Enhanced AAL2 Extended Parameter RAM

See description in Table .

41.7 User Defined Cells in AAL2

The user defined cell (UDC) mode for ATM, see Section 32.2.16, “User-Defined Cells (UDC),” applies generally to AAL2 operation as well. However, for AAL2 operation only, the UDC headers reside in a table in external memory, not in the BDs. This table resides in the same bus as the data buffers, as indicated in the RCT/TCT DTB bit.

For transmit channels in AAL2 UDC mode, initialize its UDC header entry in the TX UDC header table before activating the channel. The header can be up to 12 bytes. The TX_UDC_Base parameter in the parameter RAM, see Table 32-18, points to the beginning of the TX UDC header table.

The UDC header of a specific AAL2 transmit VC is located at the following address:

TX_UDC_Base + CH# *16 (where CH# is the ATM channel number)

For receive channels in AAL2 UDC mode, the receiver copies the UDC header from the first cell received by the VC to the RX_UDC header table. The UDC headers of subsequent cells of that VC are discarded; UDC extended address mode (UEAD) is not affected.

The UDC header of a specific AAL2 receive VC is located at the following address:

RX_UDC_Base + CH#*16 (where CH# is the ATM channel number)

The structure of a UDC header table (receive or transmit) is shown in Figure 41-33.

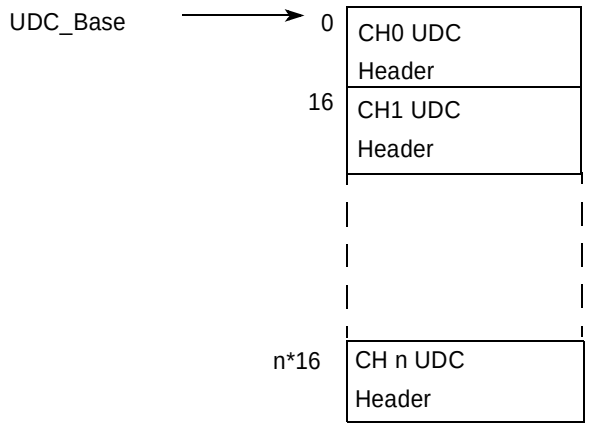


Figure 41-33. UDC Header Table

41.8 AAL2 Exceptions

For each VC, four circular interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ] for each VC, the user assigns an interrupt queue number.

When one of the CIDs generates an interrupt request, the QUICC Engine block writes a new entry to the interrupt queue containing the ATM channel number, the CID, and a description of the exception. Because CID = 0 is a unique CID number, it is used to specify that the event is related to the VC rather than the CID. As with all ATM exceptions, the valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches a location with the W bit set, it wraps to the first entry in the queue. More details can be found in Section 32.3.13.2, “Interrupt Queues.”

An interrupt entry for a CID is shown in Figure 41-34.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**V**	—	**W**	CID								TBNR	RXB	BSY	TXB	RXF
Offset + 0x02	Channel Code (CC)															

Figure 41-34. AAL2 Interrupt Queue Entry CID ≠ 0

Table 41-19 describes the interrupt queue entry fields for a CID.

Table 41-19. AAL2 Interrupt Queue Entry CID ≠ 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the QUICC Engine block. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. The exception occurred for this CID.
	11	TBNR	Tx buffer not ready interrupt. This interrupt is issued when the QUICC Engine block tries to open a TxBD, which is not ready (R = 0). This interrupt is sent only if TxQD[BNM] = 1. The interrupt has an associated channel code. **Note:** The CID number which is placed in the interrupt queue in case of TBNR is always set to 0, since the CID is not updated as the BD is not ready.
	12	RXB[1]	Rx buffer interrupt. This interrupt is issued when the I bit is set for an RxBD and the RxQD[RBM] bit is set. This interrupt has an associated channel code and CID.
	13	BSY	Busy condition. The RxBD table associated with this channel's CID is busy. Packets were discarded due to this condition.
	14	TXB	Transmit buffer interrupt. This interrupt is issued when the TxBD[I] bit is set. This interrupt is sent only if TxQD[TBM] is set. This interrupt has an associated channel code and CID.
	15	RXF[1]	Receive SSSAR SDU (frame). An SSSAR frame belonging to this channel's CID has been received. This interrupt is sent only if RxQD[RFM]=1.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

[1] These interrupt queue fields are defined differently for other AAL types, see Table 32-67.

An interrupt entry for the VC is shown in Figure 41-35.

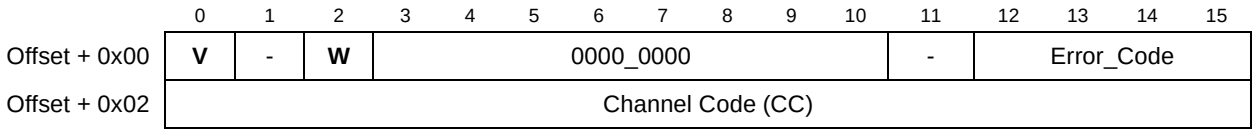


Figure 41-35. AAL2 Interrupt Queue Entry CID = 0

Table 41-20 describes the interrupt queue entry fields for the VC. All the receive error events are enabled by setting RCT[EM].

Table 41-20. AAL2 Interrupt Queue Entry CID = 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the QUICC Engine block. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	-	Reserved. Should be cleared.
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. Equals zero. This exception applies to the whole cell.
	11	-	Reserved. Should be cleared.
	12-15	Error_Code	If QM1 and QM2 are cleared: A receive error was detected. 0000 Parity error of the OSF. 0001 The STF sequence number is incorrect. 0010 The number of octets expected to overlap into this cell does not match the OSF. 0011 OSF is greater than 47. 0100 A packet HEC error was detected. 0101 The length of the CPS packet exceeds the Max_SDU_Length. 0111 A packet HEC error was detected in a split header packet.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

41.8.1 Interrupt Entry for CID Statistics

For CID Statistics Mode there is an additional interrupt entry format, that contains also a CID field. Figure 41-36 describes this Entry.

Upon getting an overflow in any of the statistics table and the Mask bit in the RCT/TCT is set an interrupt is generated.

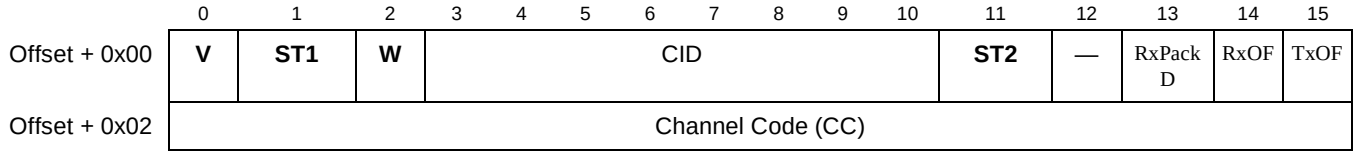


Figure 41-36. AAL2 Interrupt Entry for CID Statistics

Table 41-21 describes the interrupt queue entry fields.

Table 41-21. AAL2 Interrupt Queue for CID Statistics Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the QUICC Engine block. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	ST1	Set to 1 by the QUICC Engine block when this is a Statistics type interrupt. This gives the host a mean to distinguish this type of interrupt from other AAL2 interrupts.
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number on which the counter has overflown. This field could be a used as an index to the counters table with the exception of CID=0 on the receiver side. In this case the counter is not aligned as all the other counters in the CID mapping
	11	ST2	Set to 1 by the QUICC Engine block when this is a Statistics type interrupt. This gives the host a mean to distinguish this type of interrupt from other AAL2 interrupts.
	12	—	Reserved. Should be initialized to 0.
	13	RxPackD	Rx Packet Discard counter overflow. Valid Only for Switch mode with TxQD[PackD] bit set. This interrupt is sent only if RCT[StatsIE] is set.
	14	RxOF	Rx CID / ATM Cell counter overflow interrupt. If the CID field equals to 0, then it indicates that it is a Rx ATM Cell counter overflow, else it indicates the CID value in which the Rx CID counter overflow appears.This interrupt is sent only if RCT[StatsIE] is set.
	15	TxOF	Tx CID / ATM Cell counter overflow interrupt. If the CID field equals to 0, then it indicates that it is a Tx ATM Cell counter overflow, else it indicates the CID value in which the Tx CID counter overflow appears. This interrupt is sent only if TCT[StatsIE] is set.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

41.8.2 Interrupt Entry for WFQ Queue Management

For WFQ Queue Management mode there is an additional interrupt entry format, that it's CID field is cleared. Figure 41-37 describes this Entry.

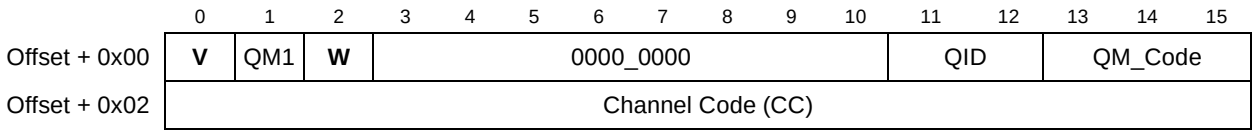


Figure 41-37. AAL2 Interrupt Queue Entry CID = 0 for WFQ Queue Management

Table 41-22 describes the interrupt queue entry fields for the VC. All the error events are enabled by the corresponding CPS_TxQD[Queue_Management] interrupt mask bits. See Table 41-6.

Table 41-22. AAL2 Interrupt Queue Entry CID = 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the RISC. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	QM1	When set identifies that it is a Switch queue management threshold event.
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. Equals zero.
	11-12	QID	Queue ID. Indicates under which queue ID the queue management interrupt occurred.
	13-15	QM_Code	Queue Management Code. 000- Full Speed to Decelerate interrupt. 001- Res. (Not used so we can identify between statistic cell counter interrupt TxOF) 010- Res. (Not used so we can identify between statistic cell counter interrupt RxOF) 011- Decelerate to Full Stop interrupt. 100- Accelerate to Decelerate interrupt. 101- Full Stop to Accelerate interrupt. 110- Accelerate to Full Speed interrupt. 111- Decelerate to Accelerate interrupt. For more description see Section 41.4.3.1.1, “Queue Management”.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

41.9 AAL2 RAM Usage

Table 41-23 summarize the AAL2 RAM usage in multithread (mth) or non-multithread (non-mth) systems.

Table 41-23. AAL2 RAM Usage

Parameter	UCCs (If MTH disabled)	Threads (If MTH Enabled)	VCCs	AVCON	WFQ	CPS	SW/SW Type1	SSSAR	Threshold no.	Bytes
Parameter RAM	1-8	1-32	—	—	—	—	—	—	—	UCCs*512 (non-mth) or UCCs*352+Threads*224 (mth)
RxQD Tmp	1-8	1-32	—	—	—	—	—	—	—	UCCs*32 (non-mth) or Threads*32 (mth)
EAAL2_PAD_TMP_BASE	—	—	—	—	—	—	—	—	—	64
ATM_AVCON_BASE	—	—	1-64K	V	—	—	—	—	—	VCCs/8 (only if AVCON)
PPRS_INT_PTR	1-8	1-32	—	—	—	—	—	—	—	UCCs*64 or Threads*64

Table 41-23. AAL2 RAM Usage (continued)

Parameter	UCCs (If MTH disabled)	Threads (If MTH Enabled)	VCCs	AVCON	WFQ	CPS	SW/SW Type1	SSSAR	Threshold no.	Bytes
Thresholds_Base	1-8	—	—	—	—	—	—	—	1-256	UCCs*8*thresholds no.
TxCIDStatsOffsetBase *	—	—	1-64K	—	—	—	—	—	—	VCCs*(2+512)
CID_MAPPING_BASE *	—	—	1-64K	—	—	—	—	—	—	VCCs*1028 - statistics enabled. VCCs* 512 - statistics disabled.
WFQ Management Table	—	—	1-64K	—	—	—	—	—	—	VCCs*32
Int RxQDs*	—	—	1-64K	V	—	1-256 ***	1-256***	1-256***	—	AVCON dis-VCCs*(8*CPS+4*SW+8*SWType1+32=*SSSAR) AVCON En-VCCs*(8*CPS+8*SW+12*SWType1+32*SSSAR)
Int TxQDs *	—	—	1-64K	—	V	1-9 **	1-9 or 1-4 (WFQ En)	1-9	—	VCCs*(16*CPS+32*SW+32*SSSAR)

* these parameters are unlikely to reside in RAM in overloaded system

** no. of TxQDs is actually unlimited, but 9 seems to be a good upper limit.

*** The total no. of RQDs under a VCC is not bigger than 256.

Chapter 42
QMC (QUICC Multi-Channel Controller)

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. 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 42-1 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 is separated to ensure that only one TDM accesses the UCC at any given time.

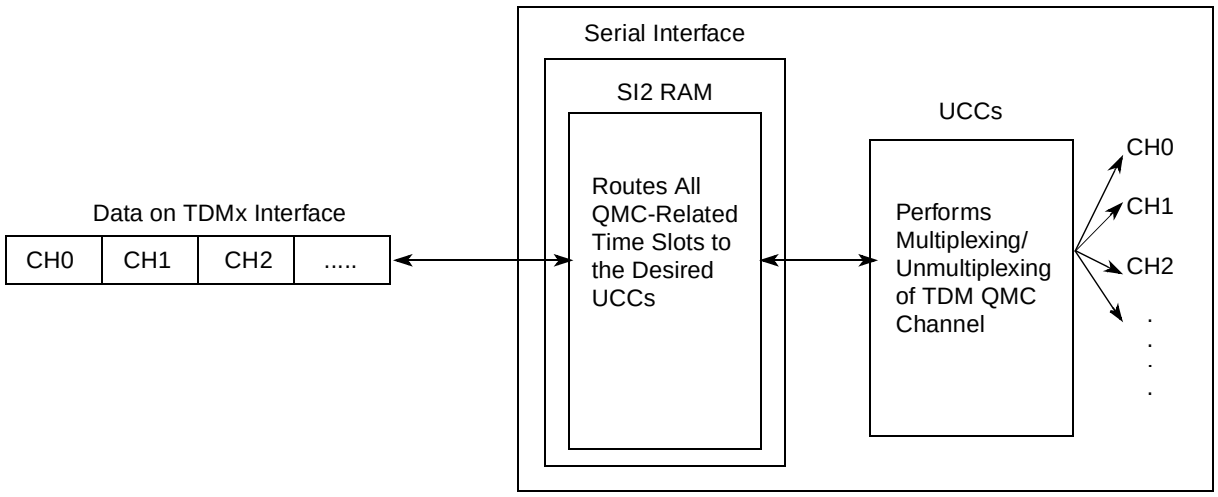


Figure 42-1. QMC Channel Addressing Capability

42.1 Features

QMC-specific features include the following:

- Up to 64 independent communication channels per UCC or split across multiple UCCs
- 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
- Subchanneling on each time slot
- Allows independent transmit and receive routing, frame syncs, and clocking
- Concatenation of any, not necessarily consecutive, time slots to channels independently for receive/transmit
- 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

42.2 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 36, “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 21.5.5, “CMX UCC Clock Route Register (CMXUCR1).”

42.2.1 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.

42.2.2 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.

42.2.3 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.

42.2.4 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.

42.2.5 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.

42.3 QMC Memory Organization

42.3.1 QMC Memory Structure

Figure 42-2 shows how data is addressed by the QMC protocol. It discusses addressing the multi-user RAM to access data within the buffers.

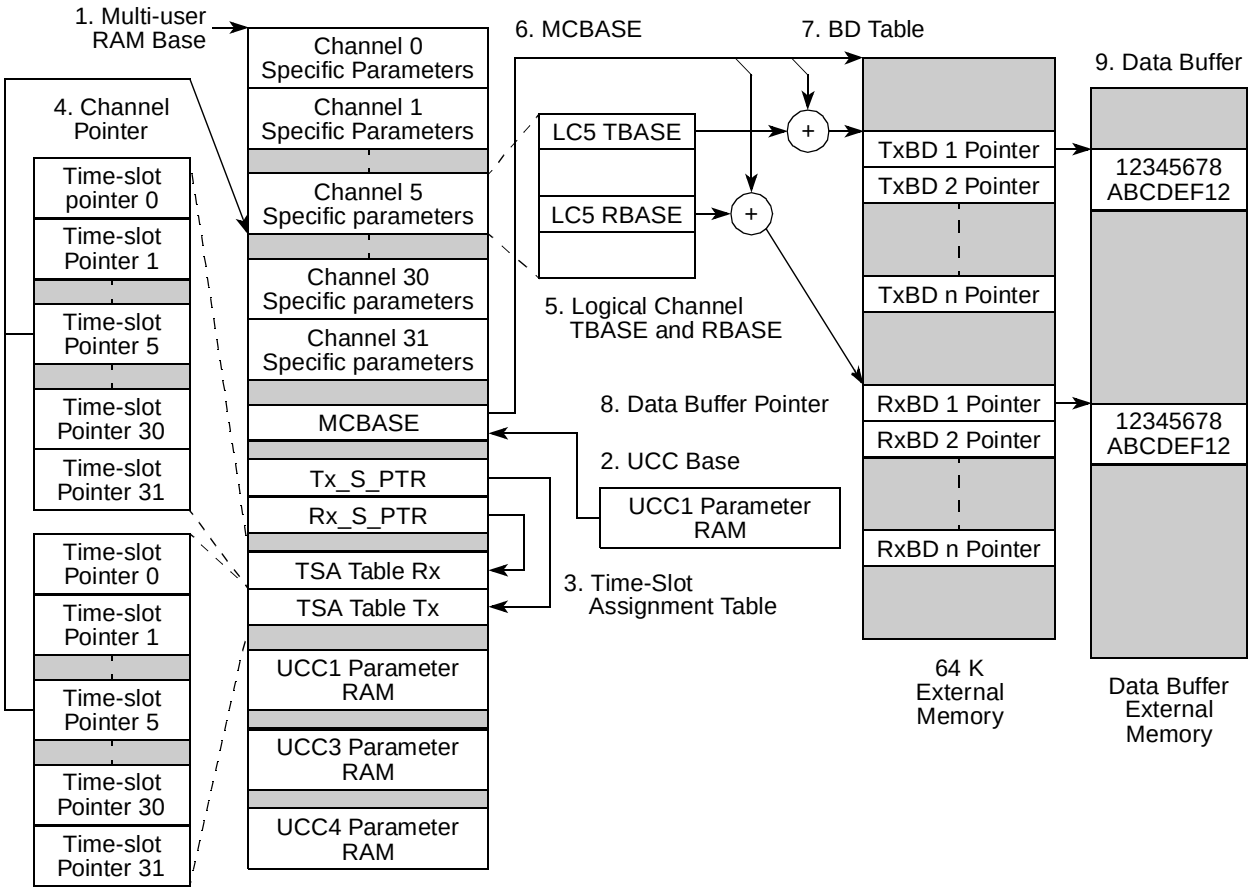


Figure 42-2. QMC Memory Structure

42.3.2 UCC Base and Global Multichannel Parameters

The UCC base points to the start of the parameter RAM for each UCC at 256-byte intervals. 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.

42.3.2.1 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 + 20; this is the normal location of the receive time slot assignment table. The Tx_S_PTR is normally

set to UCC Base + 60; this is the normal location of the transmit time-slot assignment table. However, if the receiver and the transmitter have the same mapping for the logical channels, Tx_S_PTR can point to UCC base + 20 so that Rx and Tx have a common time-slot assignment table. Note that if a single TDM channel is routed to more than one UCC, they may also use just one time-slot assignment table for all UCCs. See Section 42.3.3, "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 subchanneling, a valid bit, and a logical channel pointer. For 64-channel support there is only space for one table; therefore, common Rx and Tx parameters 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.

42.3.2.2 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 42.3.2.1, "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 42-2, 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.

42.3.2.3 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.

42.3.2.4 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.

42.3.2.5 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.

42.3.2.6 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.

42.3.2.7 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.

42.3.2.8 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 42.3.3, "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 42-1 lists the global parameters. Note that the boldfaced parameters must be initialized by the user. See Section 42.7, "QMC Initialization," for more information.

Table 42-1. Global Multichannel Parameters

Offset to UCC Base	Name	Width (Bits)	Description
00	**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.
04	**QMCSTATE**	16	Multichannel controller state (initialize to 0x8000)—Internal QMC state machine value used by RISC processor for global state definition.

Table 42-1. Global Multichannel Parameters (continued)

Offset to UCC Base	Name	Width (Bits)	Description
06	**MRBLR**	16	Maximum receive buffer length—This host-initialized entry defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. This value must be a multiple of 4 bytes.
08	**Tx_S_PTR**	16	Tx time-slot assignment table pointer (UCC base + 60 in normal mode; UCC base + 20 for common Rx & Tx time slot assignment tables)—This global QMC parameter defines the start value of the TSATTx table. The TSATTx table in the global multichannel parameter listing starts by default at UCC base + 60. Tx_S_PTR lets the user move the starting address of this table. If the same routing and masking are used for the transmitter and receiver, the tables can be overlaid, so Tx_S_PTR can point to UCC base + 20. 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.
0A	**RxPTR**	16	Rx pointer (initialize to UCC base + 20)—This global QMC parameter is a RISC variable that points to the current receiver time slot. The host must initialize this pointer to the starting location of TSATRx. The RISC processor increments this pointer whenever it completes the processing of a received time slot.
0C	**GRFTHR**	16	Global receive frame threshold—Used to reduce interrupt overhead when many short HDLC frames arrive, each causing an RXF interrupt. GRFTHR can be set to limit the frequency of interrupts. Set to 1 to get an interrupt per frame received. Note that the RXF event is written to the interrupt table on each received frame, but GINT is set only when the number of RXF events (by all channels) reaches the GRFTHR value. GRFTHR can be changed on-the-fly. For information about exception handling, see Section 42.5, “QMC Exceptions.”
0E	**GRFCNT**	16	Global receive frame count (initialized GRFCNT = GRFTHR)—A down-counter used to implement the GRFTHR feature. GRFCNT decrements for each frame received. No other receiver interrupts affect this counter. The counter value is set to the threshold during initialization. GRFCNT is automatically reset to the GRFTHR value by the QUICC Engine block after a global interrupt.
10	**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 42.5, “QMC Exceptions.”
14	**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.
18	**Rx_S_PTR**	16	Rx time-slot assignment table pointer (default = UCC base + 20 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.
1A	**TxPTR**	16	TxPTR (initialize to UCC Base + 60)—This global parameter is a RISC variable that points to the current transmitter time slot. The host must initialize it to the starting location of TSATTx. The RISC processor increments this pointer whenever it completes the processing of a transmitter time slot.

Table 42-1. Global Multichannel Parameters (continued)

Offset to UCC Base	Name	Width (Bits)	Description
1C	**C_MASK32**	32	CRC constant (0xDEBB20E3)—Required to calculate 32-bit CRC-CCITT. C_MASK32 is written by the host during QMC initialization. It is used for 32-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 42.3.4.1, "Channel-Specific HDLC Parameters." This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
20	**TSATRx**	32 Entries x 16	Time slot assignment table Rx—Host-initialized, 16-bit-wide table with 32 entries that define mapping of logical channels to time slots for the QMC receiver. The QMC protocol looks at chunks of 8 bits regardless of whether they come from one physical time slot of the TDM or whatever other combination of bits the TSA transfers to the 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 42-4.
60	**TSATTx**	32 Entries x 16	Time slot assignment table Tx—Maps a specific logical channel to each physical time slot. Time slot assignment table Tx is a host-initialized, 16-bit table with 32 entries that define the mapping of channels to time slots for the QMC transmitter. The QMC protocol looks at chunks of 8 bits regardless if they go to one physical time slot of the TDM or whatever other combination of bits are transferred from the 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 42-4.
A0	**C_MASK16**	16	CRC constant (0xF0B8)—Required to calculate 16-bit CRC-CCITT. This constant is written by the host during QMC initialization. It is used for 16-bit CRC-CCITT calculation if HDLC mode of operation is chosen for a selected channel. (This is a programmable option. For each HDLC channel, one of two CRCs can be chosen, as programmed in CHAMR.) For more information, see Section 42.3.4.1, "Channel-Specific HDLC Parameters." This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
A4	TEMP_RBA	32	Temporary receive buffer address
A8	TEMP_CRC	32	Temporary cyclic redundancy check
AC	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.
AE	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.
B0–C3	Reserved		Should be initialized to zero
C4	**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 TSA table when the TDM functionality of the receiver is identical to that of the transmitter.

The user must program the area 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 active channels in the system. 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 42-3. RX Framer Entry

NOTE

The area between UCC base + 20 and UCC base + 9F 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 + 20; see Figure 42-5. Alternatively, another UCC parameter RAM can be used, as determined by Rx_S_PTR and Tx_S_PTR; see Figure 42-7. However implemented, the TSA tables can reside anywhere in internal memory.

Figure 42-4 shows a general 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 42-4. Time-Slot Assignment Table

Table 42-2 describes the fields in the time-slot assignment table for receive.

Table 42-2. Time-Slot Assignment Table Entry Fields for Receive Section

Field	Description
V	Valid bit—Indicates whether this time slot is valid. • 0The data in this 8-bit time slot is totally ignored and not written to any buffer. • 1The 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 (e.g. zero deletion in HDLC). Individual bits can be masked out as described later.
W	Wrap bit—Identifies the last entry in TSATRx. • 0This is not the last time slot in the frame. • 1The RISC processor wraps around and handles time slot 0 or the first 8 bits transferred from the TSA in the next request. The next request is identified by a frame synchronization pulse.
Rx channel pointer	This 6-bit field of the TSATRx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of RBASE). The 6 most-significant bits are taken from the TSATRx channel pointer field, and the 6 least-significant bits are always internally set to zero.
Mask(0–7)	Mask bits—These 8 bits identify the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. Any unmasked bit (1) is processed in the receiver for a valid time slot. Any masked bit (0) is ignored by the receiver for a valid channel and no bit counter is affected.

Table 42-3 describes the fields in the time-slot assignment table for transmit.

Table 42-3. Time-Slot Assignment Table Entry Fields for Transmit Section

Name	Description
V	• Valid bit—Indicates whether this time slot is valid. • 0Logic 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. • 1Data is transmitted from its associated buffer in combination with the mask bit settings.
W	• Wrap bit—Identifies the last entry in TSATTx. • 0This is not the last time slot in the frame. • 1The RISC processor 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 + 20. 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 + 20; see Figure 42-5.

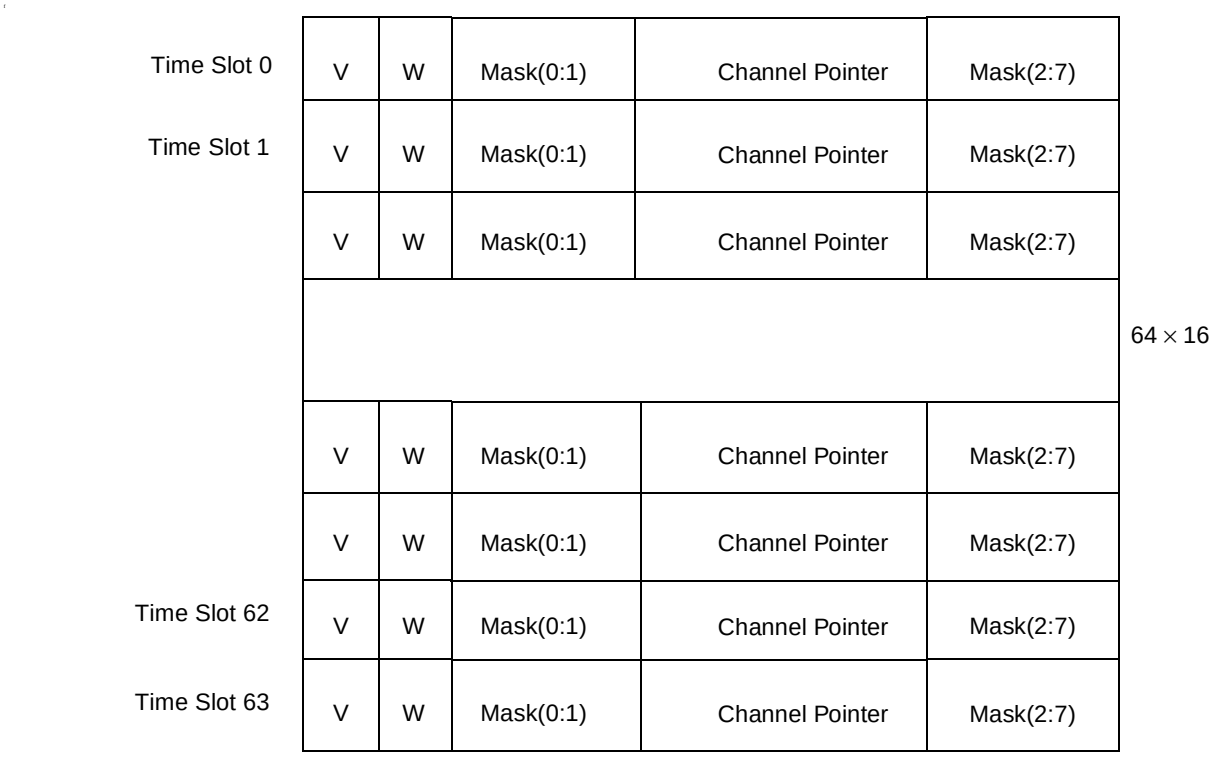


Figure 42-5. Time Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

42.3.3 Multiple UCC Assignment Tables

Assume a scenario as depicted in Figure 42-6. 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. This arrangement spreads the load over two UCCs, effectively doubling the FIFO depth on the QMC protocol. Time slots are switched to alternate UCCs to avoid data bursts that may stress the FIFOs. Each UCC sees a continuous bit stream without any gaps as described earlier.

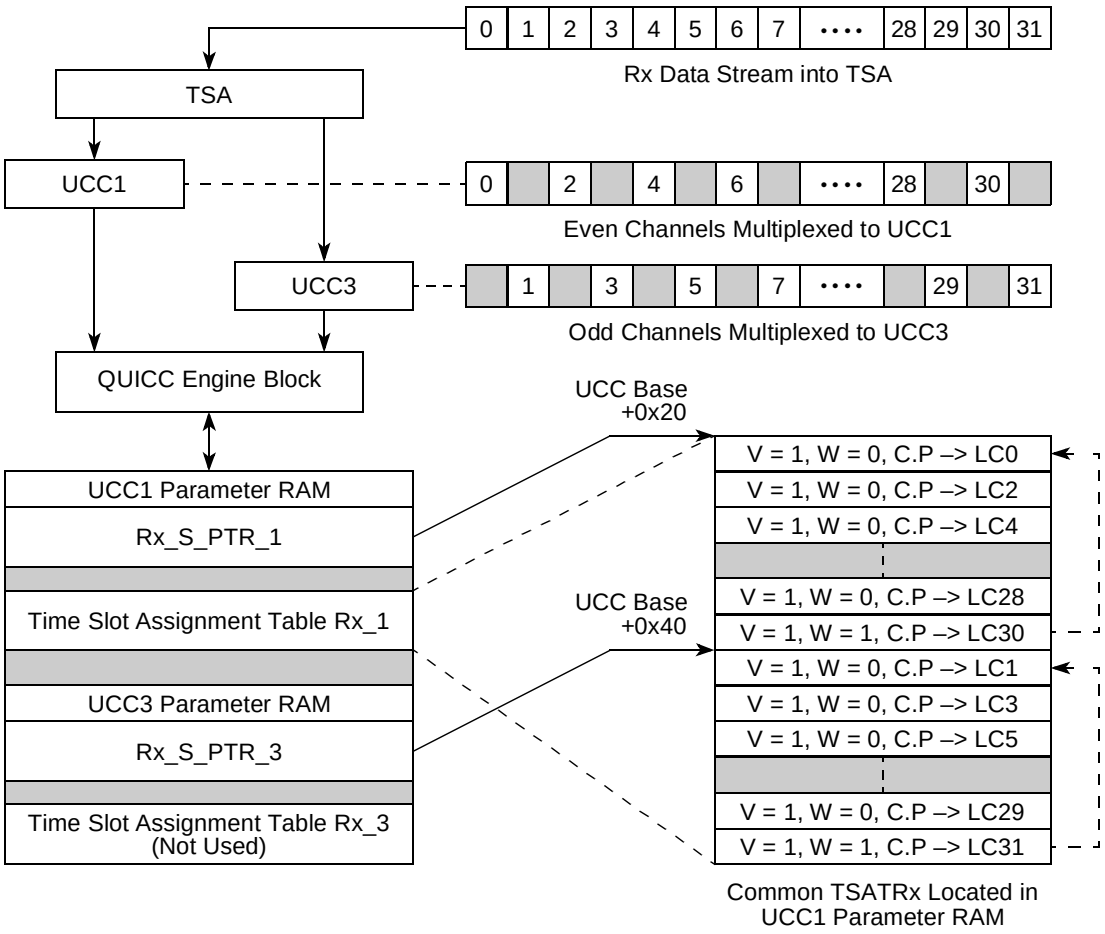


Figure 42-6. Rx Time Slot Assignment Table for 32 Channels over Two UCCs

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 42-6, 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 + 20. 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 + 40, 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 42-7. 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 to spread the load over two UCCs and facilitate separate routing for the Rx and Tx logical channels. This requires two 64-entry tables with 256 bytes, 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. The user must ensure that no collisions occur on the transmit lines from several UCCs.

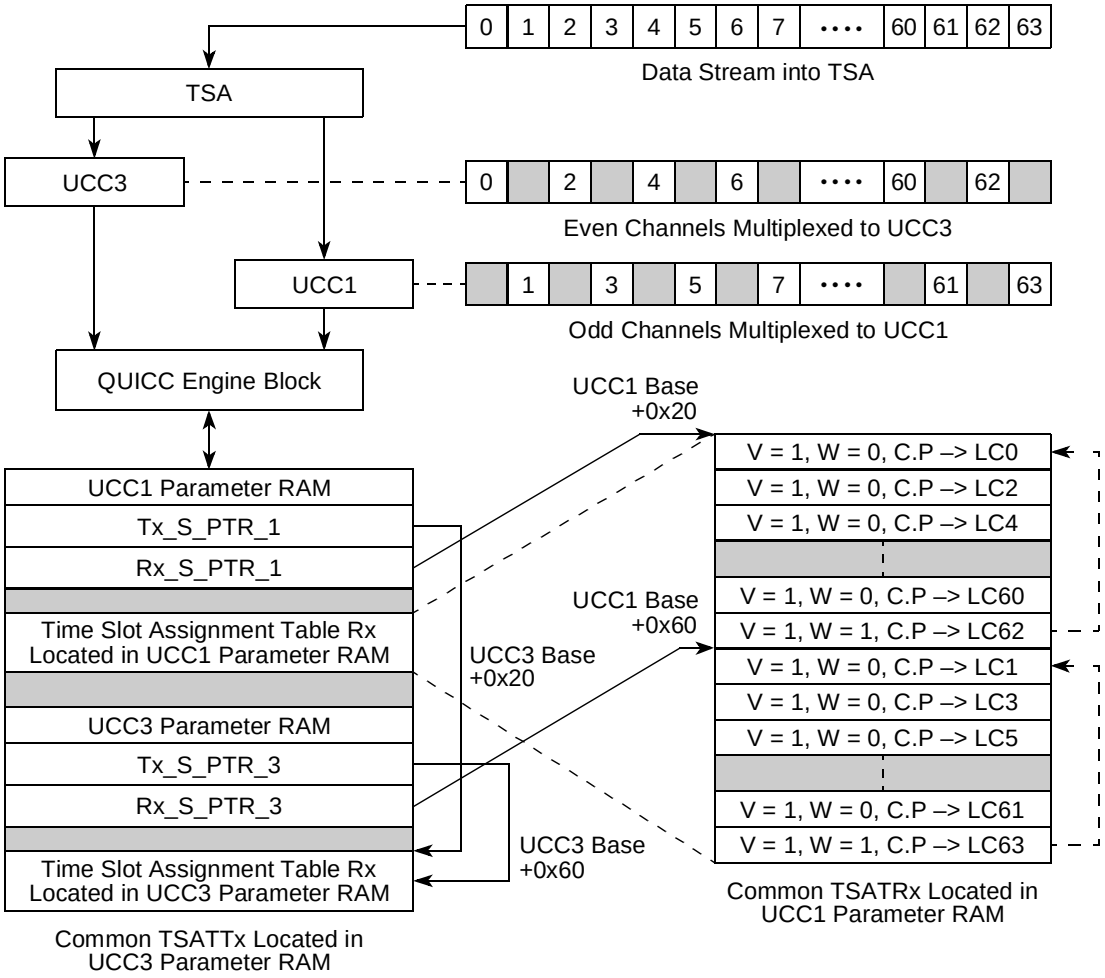


Figure 42-7. Time-Slot Assignment Tables for 64 Channels over 2 UCCs

A third scenario is depicted in Figure 42-8. A 16.384-Mbps TDM link (theoretical rate - 256 time slots of 64 Kbps) is conjugated by 4 different UCCs, each handling a group of 64 channels with its own memory space. Such a topology enables the user to apply different data contents to 256 different channels (and up to 512 in a topology with 8 UCCs).

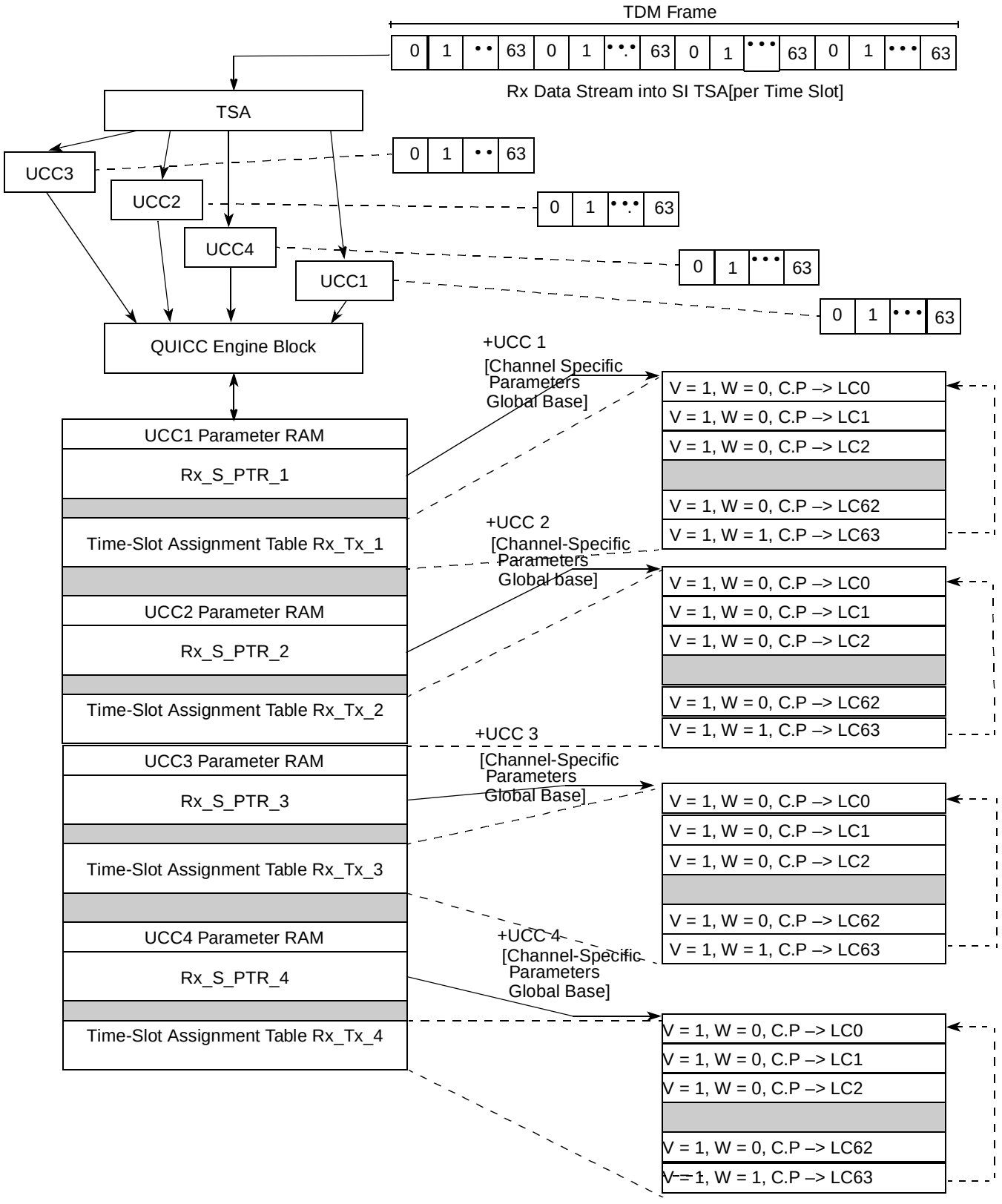


Figure 42-8. Time-Slot Assignment Tables for 256 Channels over 4 UCCs

42.3.4 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.

42.3.4.1 Channel-Specific HDLC Parameters

Table 42-4 describes the channel-specific HDLC parameters. Boldfaced parameters must be initialized by the user.

Table 42-4. Channel-Specific HDLC Parameters

Offset	Name	Width (Bits)	Description
00	**TBASE**	16	Tx buffer descriptor base address—Offset of the channel's transmit buffer descriptor table relative to MCBASE, host-initialized. See Figure 42-2.
02	**CHAMR**	16	Channel mode register. See Section 42.3.4.1.1, "CHAMR—Channel Mode Register (HDLC)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Initialize before enabling the channel. See Section 42.3.4.1.2, "TSTATE—Tx Internal State (HDLC)."
08	—	32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel again)—Offset of current BD relative to MCBASE. See Table 42-1. MCBASE + TBPTR gives the address for the BD in use.
0E	—	16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx Temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion state (host-initialized to 0x0000_0200 for HDLC or transparent operation)—Contains the previous state of the zero-insertion state machine.
18	TCRC	32	Temp transmit CRC—Temp value of CRC calculation result
1C	**INTMSK**	16	Channel's interrupt mask flags—See Section 42.3.4.1.3, "INTMSK—Interrupt Mask (HDLC)."
1E	BDFlags	16	Temp
20	**RBASE**	16	Rx buffer descriptor offset (host-initialized)— Defines the offset of the channel's receive BD table relative to MCBASE (64-Kbyte table). See Figure 42-2.
22	**MFLR**	16	Maximum frame length register (host-initialized)—Defines the longest expectable frame for this channel. Its maximum value is 64 Kbytes. The remainder of a frame which is larger than MFLR is discarded and a flag in the last frame's BD is set (LG). An interrupt request (RXF and RXB) might be generated depending on the interrupt mask. The frame length is considered to be everything between flags, including CRC. MFLR is checked every long word, but the content may be on any number of bytes. If MFLR is set to 5 for example, checking is done when 8 bytes 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.

Table 42-4. Channel-Specific HDLC Parameters (continued)

Offset	Name	Width (Bits)	Description
24	**RSTATE**	32	Rx internal state. See Section 42.3.4.1.4, "RSTATE—Rx Internal State (HDLC)," for more information.
28	—	32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Table 42-1. MCBASE + RBPTR gives the address for the BD in use.
2E	—	16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx Temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0x8000_0000.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x80FF_FFE0 in HDLC mode prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of zero deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, hold the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel. More information is given in the application note section.
38	RCRC	32	Temp receive CRC—Temp value of CRC calculation result
3C	MAX_cnt	16	Max_length counter—Count length remaining
3E	TMP_MB	16	Temp—Holds MIN(MAX_cnt, Rx internal byte count)

42.3.4.1.1 CHAMR—Channel Mode Register (HDLC)

CHAMR is a host-initialized register. Figure 42-9 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 42-9. CHAMR—Channel Mode Register (HDLC)

Table 42-5 describes the channel mode register's fields for HDLC operation. Boldfaced parameters must be initialized by the user.

Table 42-5. 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	—	0

Table 42-5. CHAMR Field Descriptions (HDLC) (continued)

Field	Name	Description
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 42.6.2, "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.
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 42-4) 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
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	—	0
10–11	—	Reserved
12–15	**NOF**	Number of flags. Defines the minimum number of flags before frames. However, even if NOF = 0, at least one flag is transmitted before the first frame. See the description of the IDLM bit for more information.

42.3.4.1.2 TSTATE—Tx Internal State (HDLC)

TSTATE defines the internal transmitter state. Figure 42-10 shows the TSTATE 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 42-10. TSTATE—TX Internal State (HDLC)

Table 42-6 describes the TSTATE fields.

Table 42-6. 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 19.1.8.2, "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 Figure 19-1 and Section 5.4.3.7, "Debug Configuration."
6	DTB	Should be cleared
7	BDB	Should be cleared

42.3.4.1.3 INTMSK—Interrupt Mask (HDLC)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK, as shown in Figure 42-12. 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 42.5, "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 42-11. Interrupt Table Entry

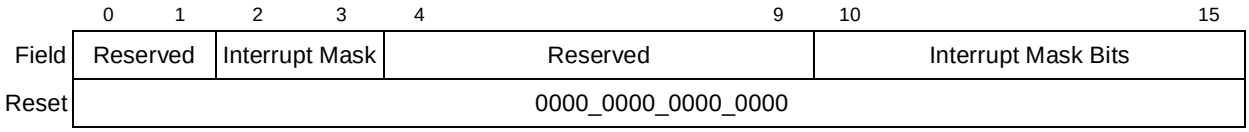


Figure 42-12. INTMSK (HDLC)

42.3.4.1.4 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 42-13 shows the RSTATE register for HDLC operation.

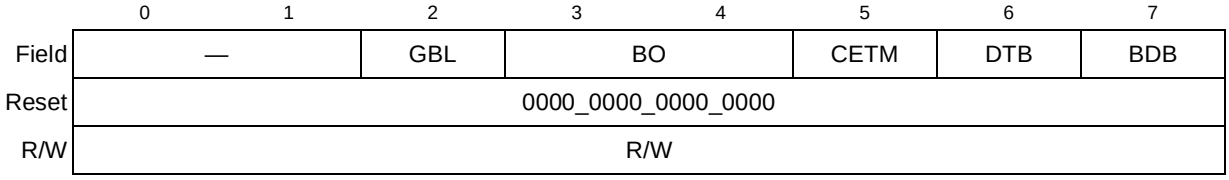


Figure 42-13. RSTATE—RX Internal State (HDLC)

Table 42-7 describes the RSTATE fields.

Table 42-7. 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 19.1.8.2, “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 Figure 19-1 and Section 5.4.3.7, “Debug Configuration.”
6	DTB	Should be cleared
7	BDB	Should be cleared

42.3.4.2 Channel-Specific Transparent Parameters

Table 42-8 describes the channel-specific transparent parameters. Boldfaced parameters must be initialized by the user.

Table 42-8. Channel-Specific Transparent Parameters

Offset	Name	Width	Description
00	**TBASE**	16	Tx buffer descriptor base address—Defines the offset of the channel's transmit BD table relative to MCBASE, host-initialized. See Figure 42-2.
02	**CHAMR**	16	Channel mode register. See Section 42.3.4.2.1, "CHAMR—Channel Mode Register (Transparent Mode)."
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Initialize before enabling the channel. See Section 42.3.4.2.2, "TSTATE—Tx Internal State (Transparent Mode)."
08		32	Tx internal data pointer—Points to current absolute address of channel.
0C	**TBPTR**	16	Tx buffer descriptor pointer (host-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel)—Contains the offset of current BD relative to MCBASE. See Table 42-1. MCBASE + TBPTR gives the address for the BD in use.
0E		16	Tx internal byte count—Number of remaining bytes
10	TUPACK	32	(Tx temp) Unpack 4 bytes from 1 long word
14	**ZISTATE**	32	Zero-insertion machine state (host-initialized to 0x0000_0200)—Contains the previous state of the zero-insertion state machine.
18	RES	32	—
1C	**INTMSK**	16	Channel's interrupt mask flags. See Figure 42-17.
1E	BDFlags	16	Temp
20	**RBASE**	16	Receive buffer descriptor base offset—Defines the offset of the channel's 64-Kbyte receive BD table relative to MCBASE. Host-initialized. See also Figure 42-2.
22	**TMRBLR**	16	Transparent maximum receive buffer length (host-initialized entry)—Defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. Note that this value must be a multiple of 4 bytes as the QMC works on long-word alignment.
24	**RSTATE**	32	Rx internal state. See Section 42.3.4.2.5, "RSTATE—Rx Internal State (Transparent Mode)," for more information.
28		32	Rx internal data pointer—Points to current address of specific channel.
2C	**RBPTR**	16	Rx buffer descriptor pointer (host-initialized to RBASE, prior to operation or due to a fatal error)—Contains the offset from MCBASE to the current receive buffer. See Figure 42-2. MCBASE + RBPTR gives the address for the BD in use.
2E		16	Rx internal byte count—Per channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0x8000_0000.
34	**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.
38	RES	32	—

Table 42-8. Channel-Specific Transparent Parameters (continued)

Offset	Name	Width	Description
3C	**TRNSYNC**	16	Transparent synchronization—In transparent mode, this register controls synchronization for single time slots or superchannel applications. See Section 42.3.4.2.4, "TRNSYNC—Transparent Synchronization."
3E	RES	16	—

42.3.4.2.1 CHAMR—Channel Mode Register (Transparent Mode)

CHAMR is a host-initialized register. Figure 42-14 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 42-14. CHAMR—Channel Mode Register (Transparent Mode)

Table 42-9 describes the channel mode register fields for transparent operation. Boldfaced parameters must be initialized by the user.

Table 42-9. 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	—	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 42-4) 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
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	RES	Reserved

Table 42-9. CHAMR Bit Settings (Transparent Mode) (continued)

Field	Name	Description
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	—	0
10–11	—	Reserved
12–15	—	0

42.3.4.4.2.2 TSTATE—Tx Internal State (Transparent Mode)

TSTATE defines the internal transmitter state. Figure 42-15 shows the TSTATE register for transparent mode.

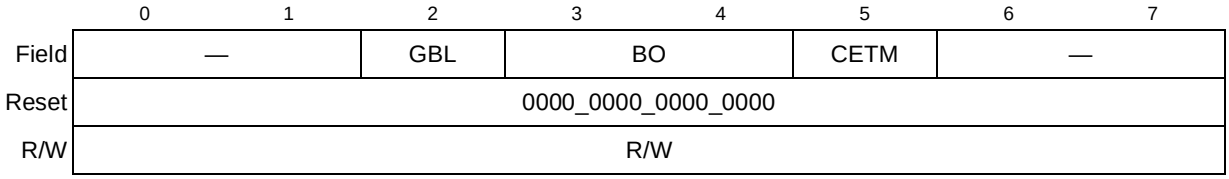


Figure 42-15. TSTATE—TX Internal State (Transparent Mode)

Table 42-10 describes the TSTATE fields.

Table 42-10. 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 19.1.8.2, “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 Figure 19-1 and Section 5.4.3.7, “Debug Configuration.”
6–7	—	Reserved, should be cleared

42.3.4.2.3 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 42-17. 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 42-16. Interrupt Table Entry

	0–10	11–15
Field	Reserved	Interrupt Mask Bits
Reset	0000_0000_0000_0000	

Figure 42-17. INTMSK (Transparent Mode)

42.3.4.2.4 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 42-18, 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:$(23 + 1) \times 2 = 0$
- C3 is a 2-byte pattern TS7, TS8, so TSn = 7
  - Rx Byte: $(7 + 1) \times 2 = 16$
  - As x = 1, TSn + x = 8, so Tx Byte:$(8 + 1) \times 2 = 18$
- C4 is a 2-byte pattern TS8, TS7, so TSn = 8
  - Rx Byte:$(8+1) \times 2 = 16$
  - As x = 1, TSn + x = 7, so Tx Byte:$(7 + 1) \times 2 = 16$
- C5 is a 2-byte pattern TS19, TS23, so TSn = 19
  - Rx Byte:$(19 + 1) \times 2 = 40$
  - As x = 1, TSn + x = 23, so Tx Byte:$(23 + 1) \times 2 = 0$
- C6 is a 2-byte pattern TS23, TS19, so TSn = 23
  - Rx Byte:$(23 + 1) \times 2 = 0$
  - As x = 1, TSn + x = 19, so Tx Byte:$(19 + 1) \times 2 = 40$
- C7 is a 4-byte pattern TS20, TS23, TS8, TS9, and TS19, so TSn = 20
  - Rx Byte:$(20 + 1) \times 2 = 42$
  - As x = 5, TSn + x = 19, so Tx Byte:$(19 + 1) \times 2 = 40$
- C8 is a 4-byte pattern TS8, TS9, TS19, TS20, and TS23, so TSn = 8
  - Rx Byte:$(8 + 1) \times 2 = 18$
  - As x = 5, TSn + x = 23, so Tx Byte:$(23 + 1) \times 2 = 0$
- C9 is a 4-byte pattern TS19, TS20, TS23, TS8, and TS9, so TSn = 19
  - Rx Byte:$(19 + 1) \times 2 = 40$
  - As x = 5, TSn + x = 9, so Tx Byte:$(9 + 1) \times 2 = 20$

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.

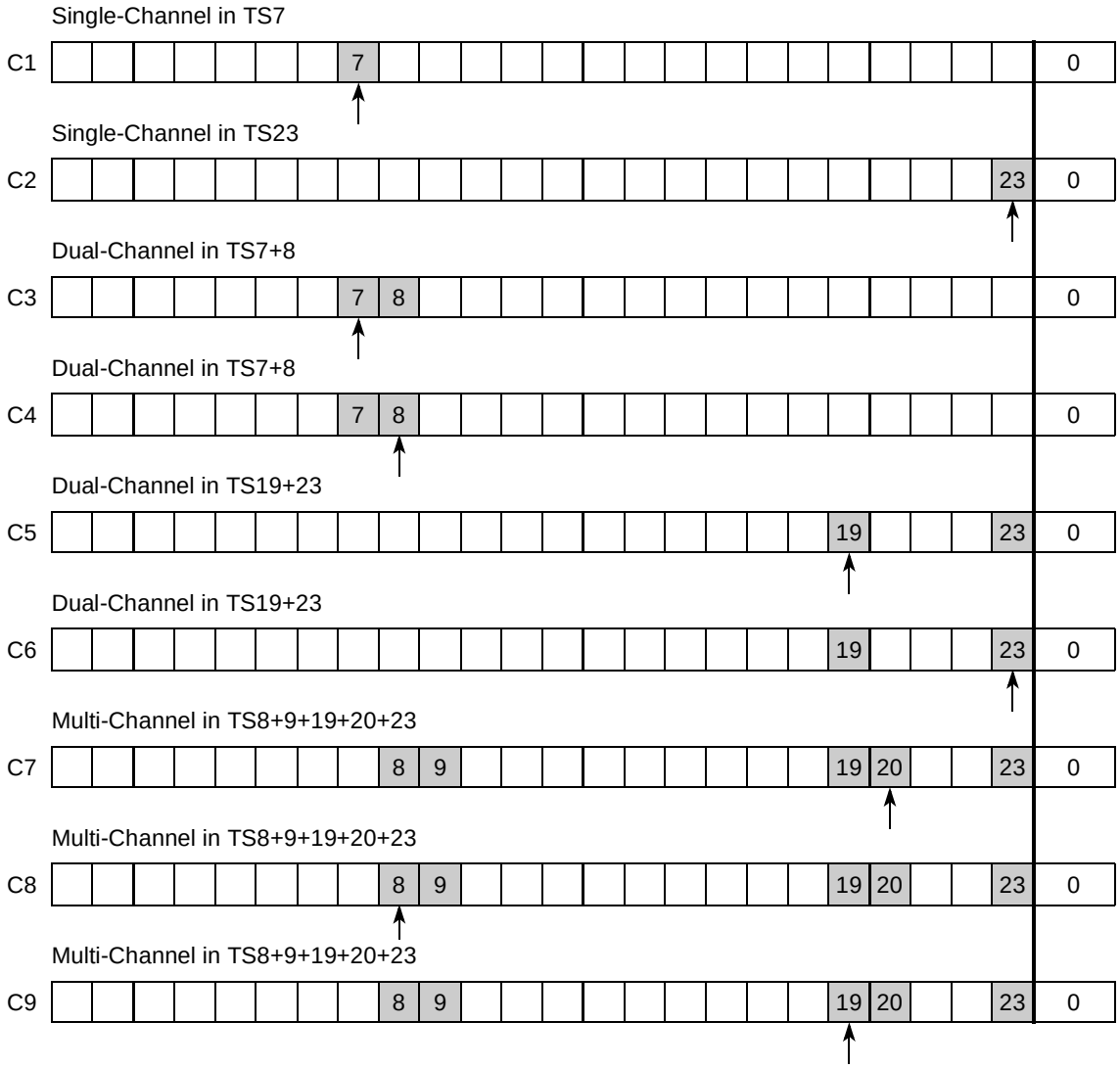


Figure 42-18. Examples of Different T1 Time Slot Allocation

42.3.4.2.5 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 42-19 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 42-19. RSTATE—RX Internal State (Transparent)

Table 42-11 describes the RSTATE fields.

Table 42-11. 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 19.1.8.2, "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 Figure 19-1 and Section 5.4.3.7, "Debug Configuration."
6–7	—	Reserved, should be cleared.

42.4 QMC Commands

The host issues commands to the QMC by writing to the command register (CECR); see Section 20.3.1, "QUICC Engine Command Register (CECR)."

42.4.1 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 42-4 and Section 42.3.2.8, "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].

42.4.2 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 GSMR register.

42.5 QMC Exceptions

QMC interrupt handling involves two principle data structures—the UCC event register (UCCE) and the circular interrupt table. Figure 42-20 illustrates the circular interrupt table.

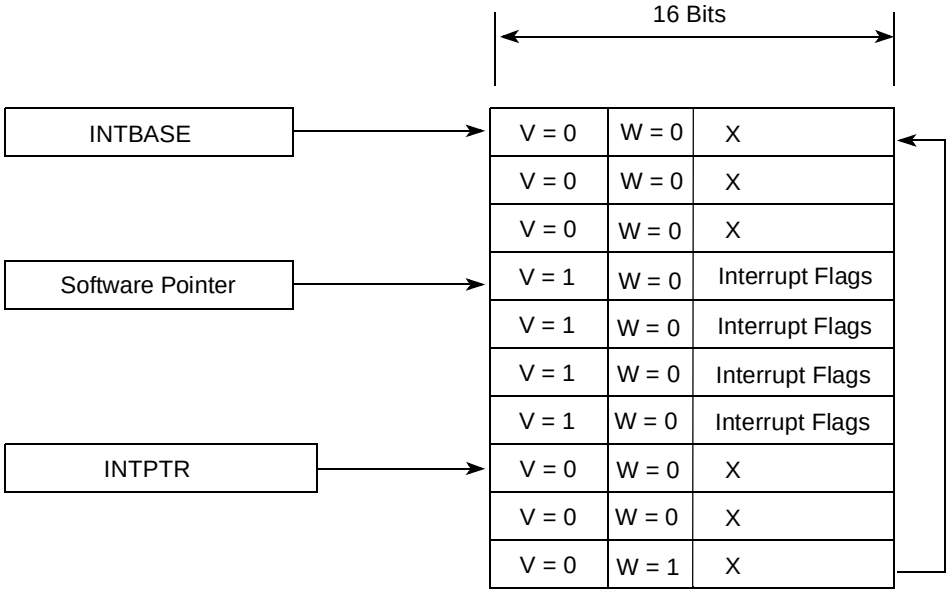


Figure 42-20. Circular Interrupt Table in External Memory

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 42-1. 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). The host follows this procedure until it reaches an entry with V = 0, indicating an invalid entry.

NOTE

It is very important that the user clear all bits but the wrap bit in a queue entry even though its valid bit may be cleared. Clearing only the interrupt bits confuse user software with incorrect channel interrupts.

42.5.1 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 42.5.3, "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 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.

42.5.1.1 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.

42.5.1.2 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.

42.5.1.3 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 42.7, "QMC Initialization."

42.5.2 UCC Event Register (UCCE)

The QMC's UCCE is used to report events and generate interrupt requests. See Figure 42-21 and Table 42-12 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 42-22 for UCCM assignments.

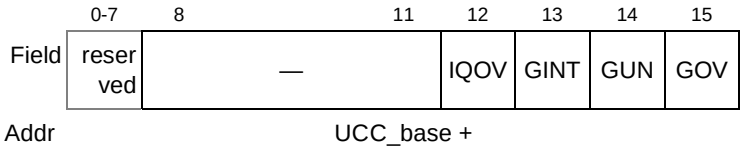


Figure 42-21. UCC Event Register

Table 42-12. 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.

	0-7	8 — 11	12	13	14	15
Field	reserved	—	IQOV	GINT	GUN	GOV
Addr	UCC_base + 0x14					

Figure 42-22. UCCM Register

42.5.3 Interrupt Table Entry

The interrupt table contains information about channel-specific events. Its flags are shown in Figure 42-23. 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 42.3.4, "Channel Specific Parameters." Table 42-13 describes the fields of an interrupt table entry.

	0	1	2	3	4 — 9	10	11	12	13	14	15
Field	V	W	NID	IDL	Channel Number	MRF	UN	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000										

Figure 42-23. Interrupt Table Entry

Table 42-13. 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).

Table 42-13. Interrupt Table Entry Field Descriptions (continued)

Field	Name	Description
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.
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 42.7.1, "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 42.7.2, "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 42.6, "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.

42.5.4 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.

42.5.5 Channel Interrupt Processing Flow

Figure 42-24 illustrates the flow of a channel interrupt. Note that this does not describe the processing of the global interrupts GUN and GOV.

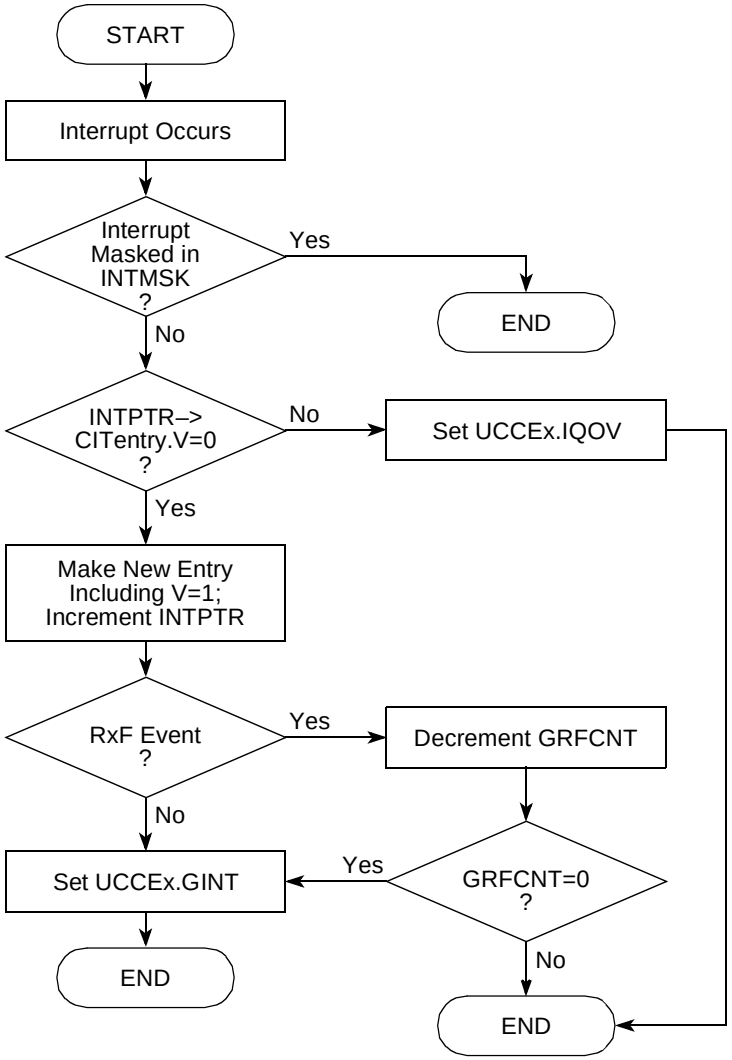


Figure 42-24. Channel Interrupt Flow

42.6 Buffer Descriptors

QMC buffer descriptors are located within 64 Kbytes in external memory; see Figure 42-2. Each buffer descriptor contains key information about the buffer it defines.

The first two sections describe the contents of the receive and transmit buffer descriptors for the QMC protocol.

The third section discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.

NOTE

The 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.

42.6.1 Receive Buffer Descriptor

Figure 42-25 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 42-25. Receive Buffer Descriptor (RxBD)

Table 42-14 describes the individual fields of a receive buffer descriptor. Boldfaced entries must be initialized by the user.

Table 42-14. 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 42-14. 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 42-26 shows how non-octet alignment is reported and how data is stored. The two diagrams on the left show the reception of a single-buffer, 12-byte frame including the CRC. In the top case, the reception is correctly octet-aligned and the frame length indicates 12 bytes.

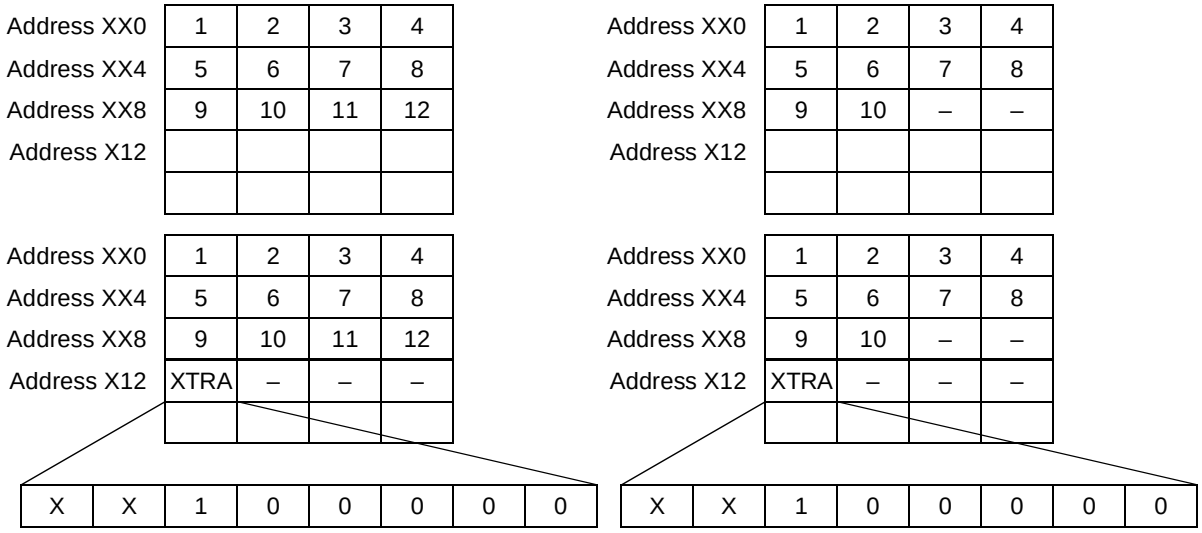


Figure 42-26. Nonoctet Alignment Data

In the bottom case, two more bits are received. The frame length is now 13 bytes, and the address positions X13 through X15 point to invalid data. Address position X12 contains information about the non-octet alignment. Valid information is written starting at the MSB position, shown as ‘x’ in the diagram. Starting from the LSB position, zeros are filled in followed by a ‘1’ immediately preceding the valid data.

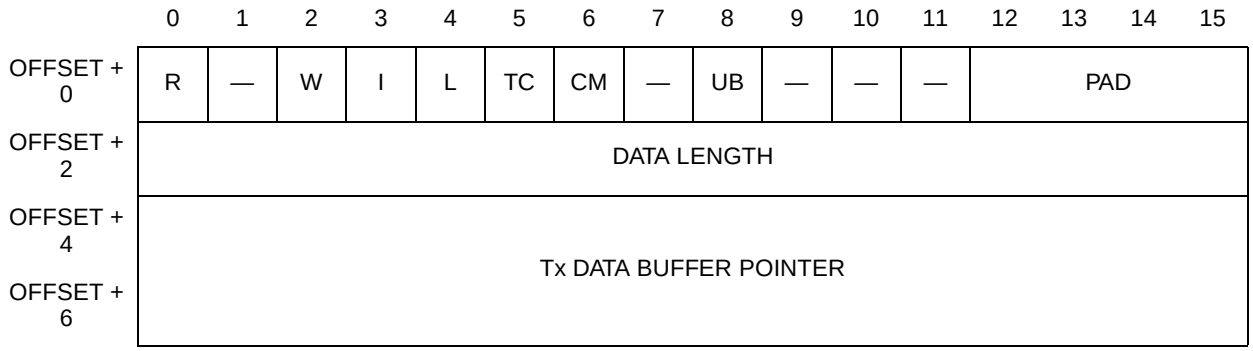
The two diagrams on the right show how the data and the extra information is stored for a frame length that is not a multiple of 4 bytes. The additional information is always on a long-word boundary. In the top case the frame length is 10 bytes and in the bottom case the length is 11 bytes.

For a minimum frame consisting of ‘flag, 1 bit, flag’ the frame length is 1. The only valid entry is at address XX0 with content of x1000000.

To accommodate the extra long word that may be written at the end of a frame and to avoid overwritten memory beyond the end of a buffer, it is recommended to reserve MRBLR + 8 bytes for each buffer area. The XTRA information shown in Figure 42-26 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.

42.6.2 Transmit Buffer Descriptor

Figure 42-27 shows the transmit buffer descriptor.



Notes: Entries in boldface must be initialized by the user.

Figure 42-27. Transmit Buffer Descriptor (TxBD)

Table 42-15 describes the individual fields of a transmit buffer descriptor. Boldfaced entries must be initialized by the user.

Table 42-15. 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.
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.

Table 42-15. Transmit Buffer Descriptor (TxBD) Field Descriptions (continued)

Field	Name	Description
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	—	—
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 42-28.
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 42-28 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.

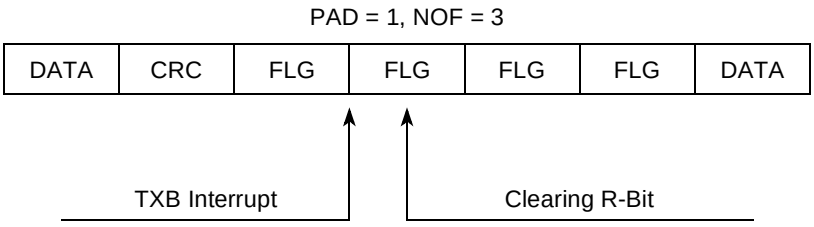


Figure 42-28. Relation between PAD and NOF

42.6.3 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.

42.6.3.1 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 42.3.2.1, “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.

42.6.3.2 Internal Memory Structure

For details about the internal memory, refer to Chapter 2, “Memory Map.”

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 free for buffer descriptors for any UCC. QMC channels can also use this space instead of external memory for buffer descriptors, reducing the load on the external bus.

42.7 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 20.3.1, “QUICC Engine

Command Register (CECR)," and GUMR_L[MODE], see Section Figure 24-1., "GUMR_L—General UCC Mode Register Low Order," 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. Initialize UCC registers, including enabling the UCC. Program GUMR_H[16] to a zero to select QMC mode, see Table 24-3.
7. Program RX and TX SIRAM to point appropriate time slots to the UCC used for QMC.
8. Initialize SI registers
9. Enable TDM

42.7.1 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.

42.7.2 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.

42.7.3 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 42.4.1, "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 42.4.2, "Receive Commands," and perform a restart as described above.

Chapter 43
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.

43.1 Features

The IMA ATM Forum Specification defines the functions shown in Table 43-1.

Table 43-1. 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

43.1.1 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

43.1.2 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.

43.1.3 PHY-Layer Devices Supported

The IMA microcode is primarily targeted at ATM's primary application (i.e. 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.

43.1.4 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) (i.e. cells that are not 53 bytes and/or with customized headers)
- Internal rate mode for APC scheduling

43.2 IMA Protocol Overview

This section describes the IMA protocol, not this specific implementation of IMA microcode.

43.2.1 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 43-1 illustrates the ATM inverse multiplexing technique in one direction. This technique applies in the opposite direction.

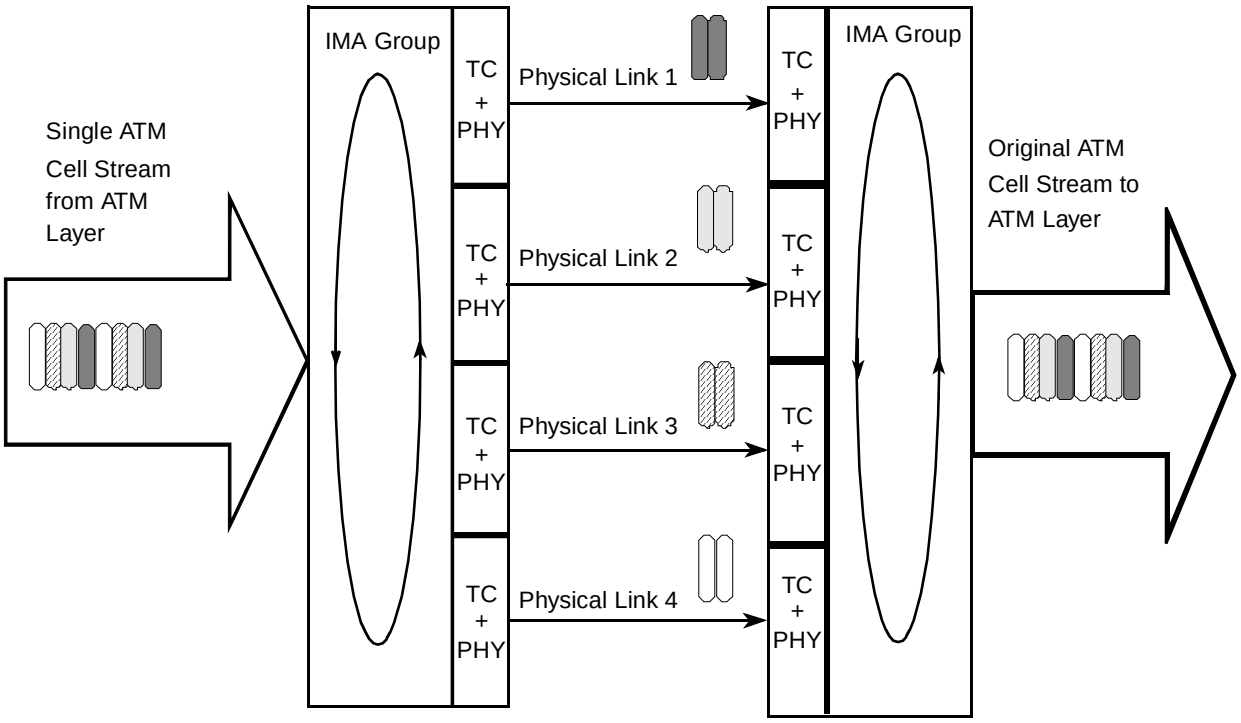


Figure 43-1. Basic Concept of IMA

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.

43.2.2 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 43-2) 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.

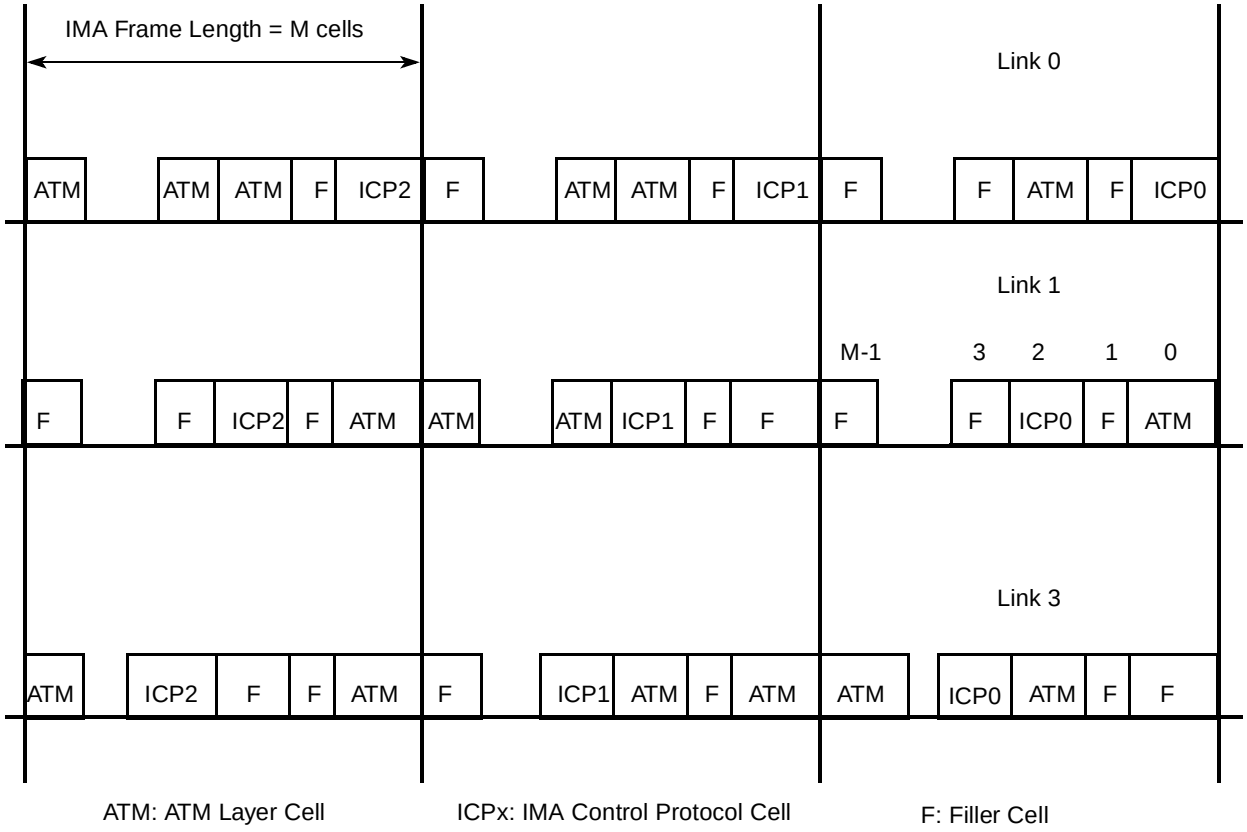


Figure 43-2. Illustration of IMA Frames

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.

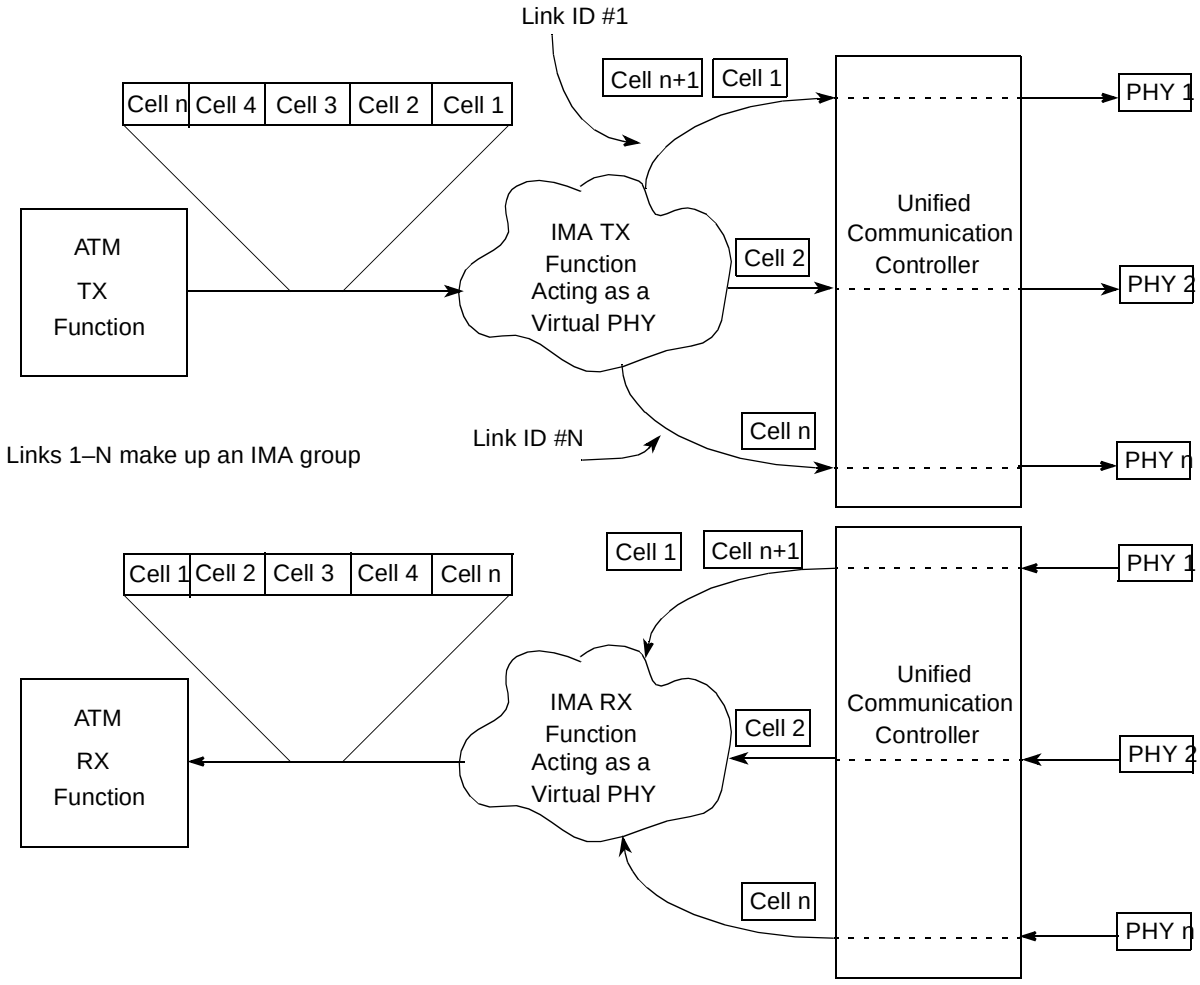


Figure 43-3. IMA Microcode Overview

43.2.3 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.

43.2.3.1 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.

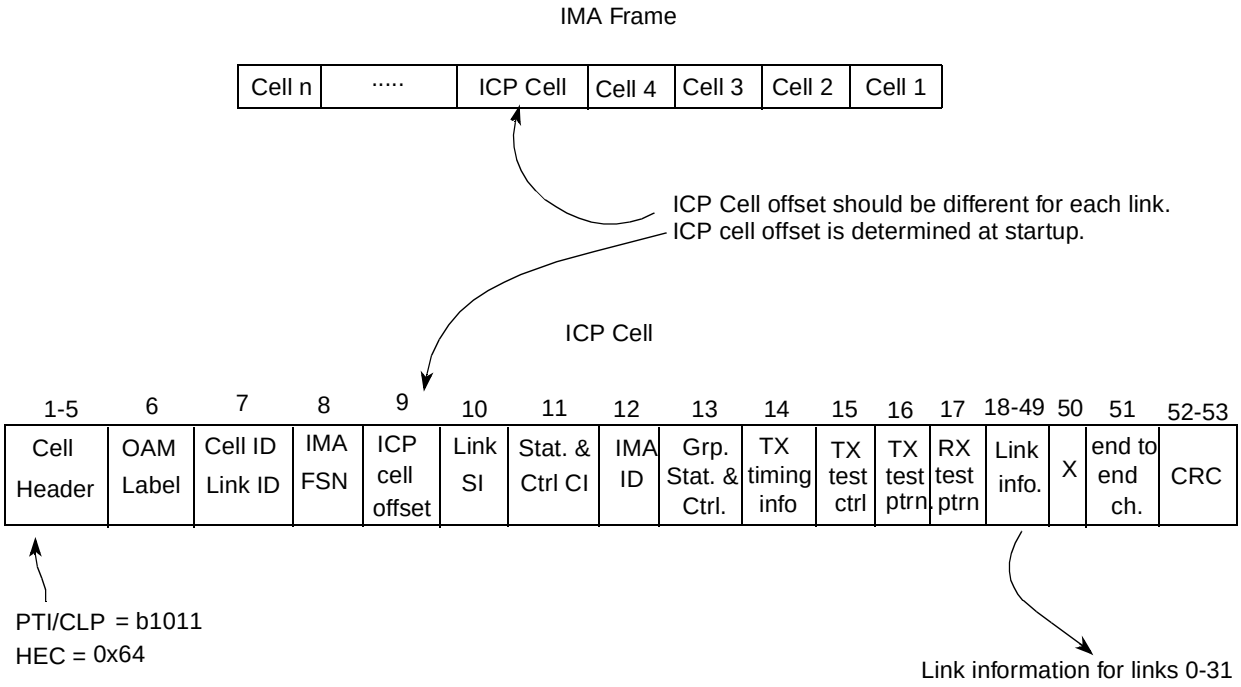


Figure 43-4. IMA Frame and ICP Cell Formats

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.

43.2.3.2 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.

43.3 IMA Microcode Architecture

This section explains the architecture of the receive and transmit IMA microcode tasks.

43.3.1 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.

43.3.1.1 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

43.3.1.2 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

43.3.2 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.

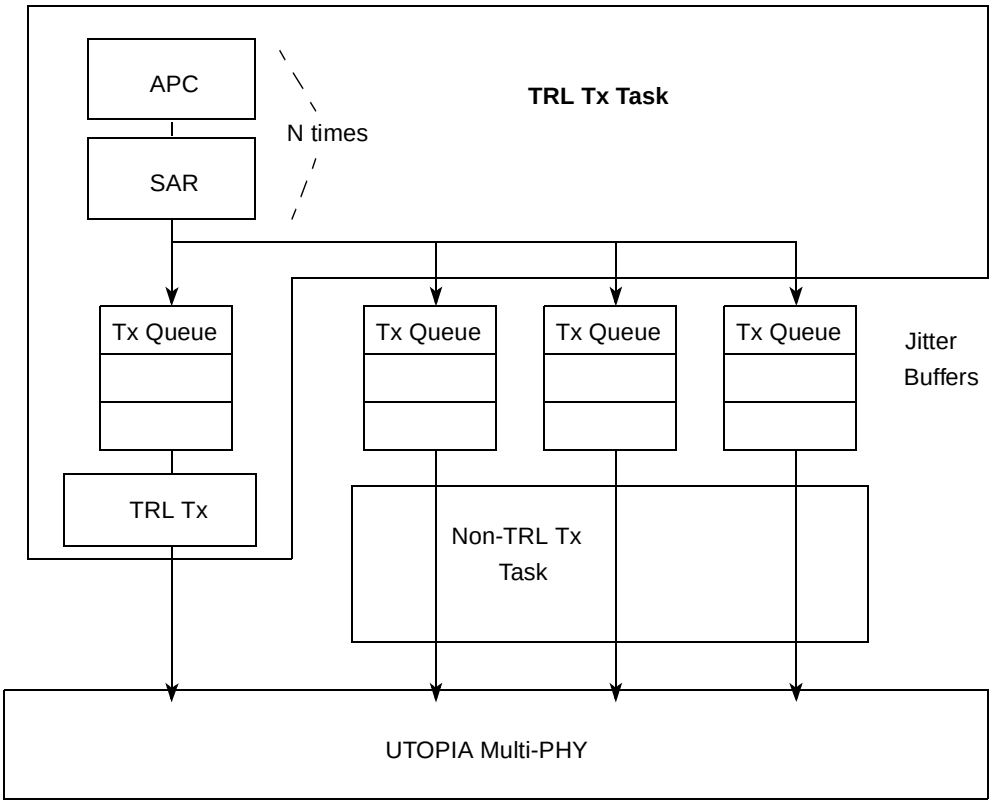


Figure 43-5. IMA Transmit Task Interaction

43.3.2.1 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.

43.3.2.1.1 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.

43.3.2.2 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.

43.3.2.3 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.

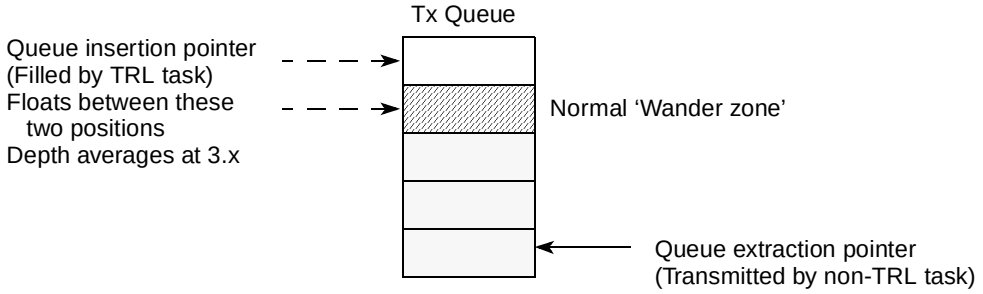


Figure 43-6. Transmit Queue Normal Operating State

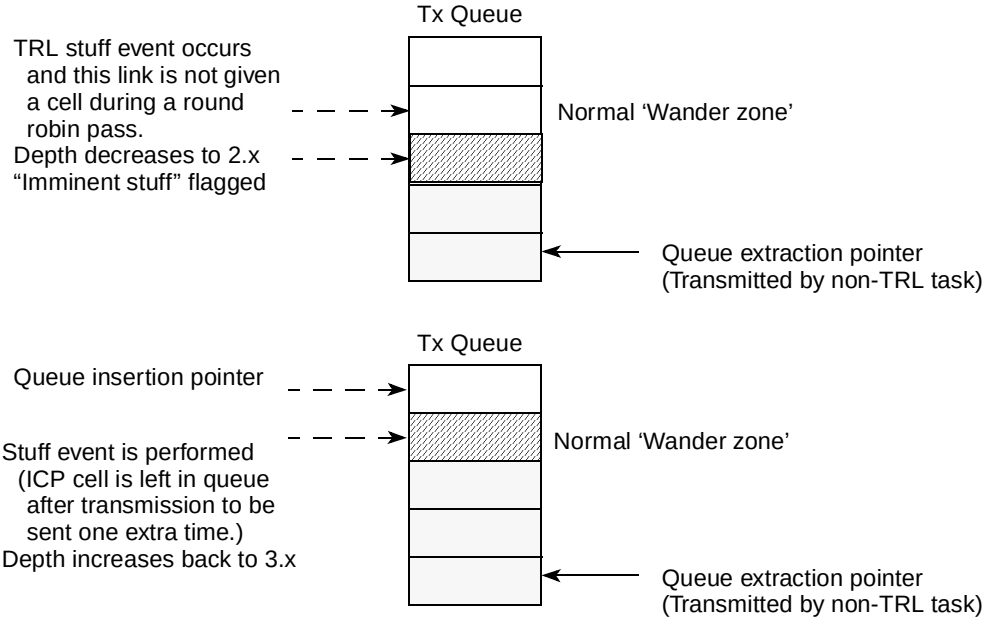


Figure 43-7. Transmit Queue Behavior: Link Clock Rate Same as TRL

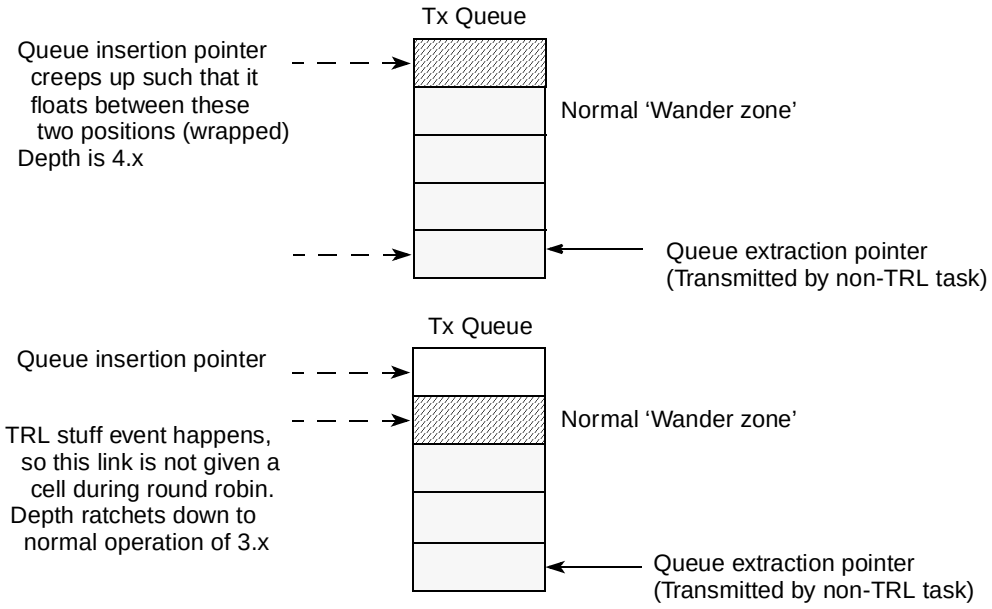


Figure 43-8. Transmit Queue Behavior: Link Clock Rate Slower Than TRL

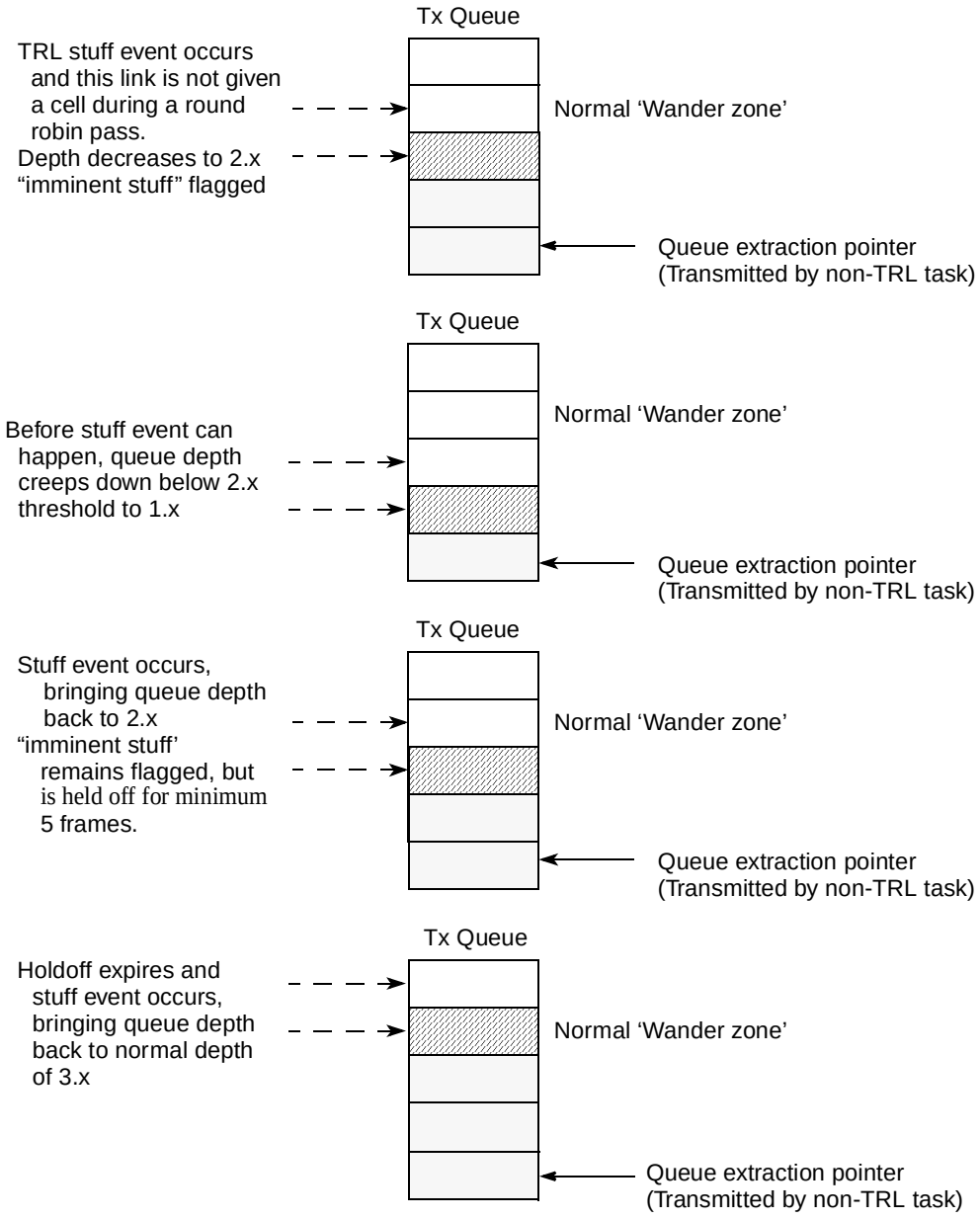


Figure 43-9. Transmit Queue Behavior: Link Clock Rate Faster Than TRL Worst-Case Event Sequence

43.3.2.4 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).

43.3.3 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.

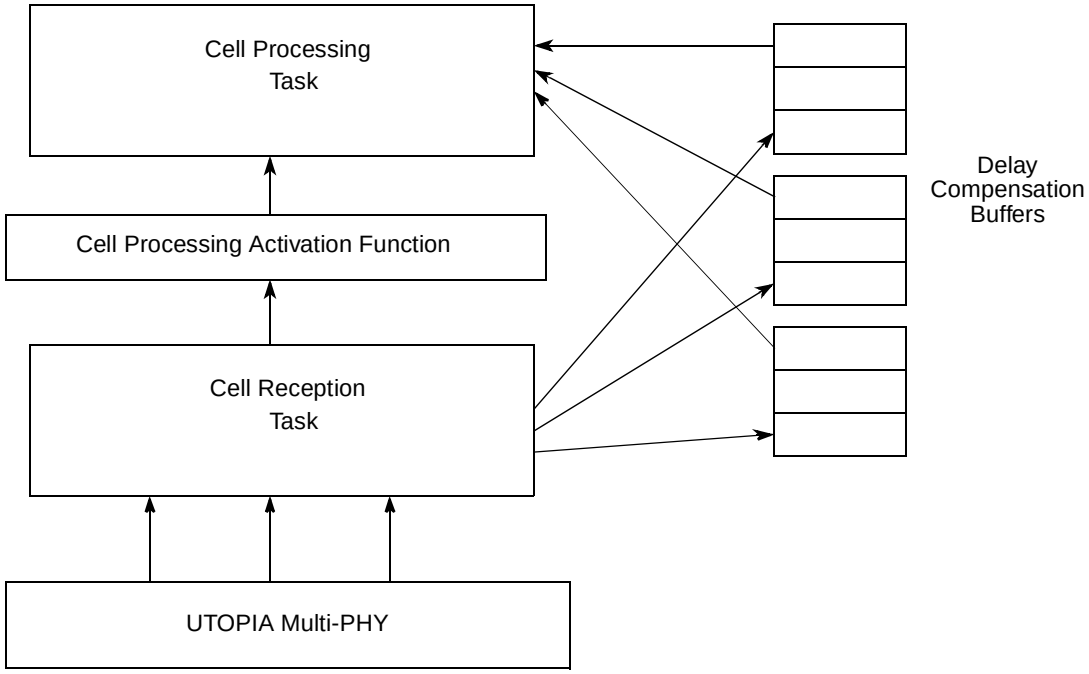


Figure 43-10. IMA Receive Task Interaction

43.3.3.1 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.

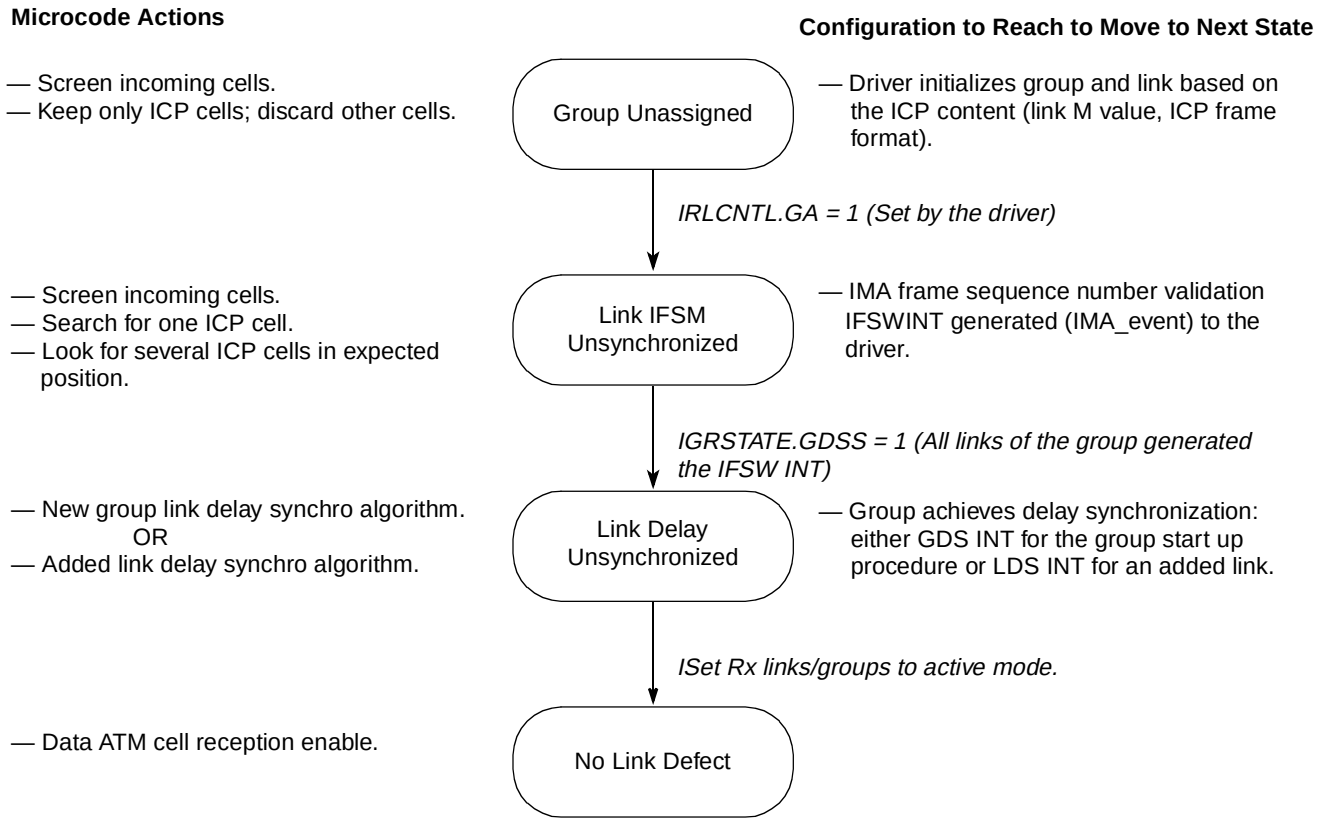


Figure 43-11. IMA Microcode: Receive Process

As Figure 43-11 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.

43.3.3.2 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.

43.3.3.3 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.

43.4 IMA Programming Model

43.4.1 Data Structure Organization

Figure 43-12 shows the organization of IMA data structures.

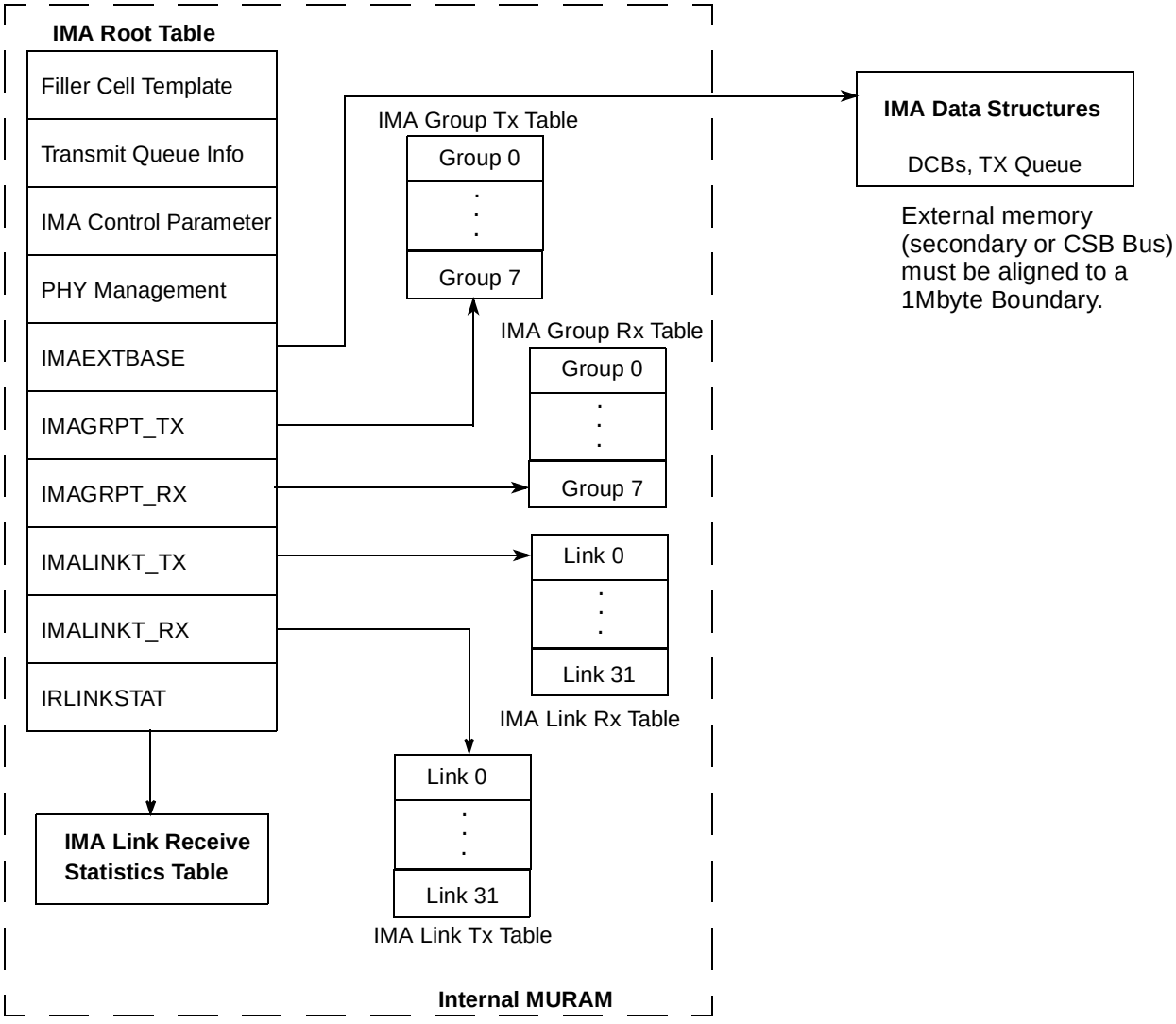


Figure 43-12. IMA Root Table Data Structures

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

43.4.2 IMA UCC Programming

43.4.2.1 UCC Registers

The UCC protocol-specific mode register (UPSMR) for ATM operation is described in Section 33.6.5.1, "UPDCx in ATM Protocol." Refer to that section for information pertaining to IMA.

For the UPC and therefore for an external TC layer of any PHY programmed in IMA mode (that is, the corresponding bit in IMA PHY is set), the corresponding UTIRRx must be programmed to disable the internal rate for this PHY. ALL IMA PHYs must operate in external rate mode. Refer to Section 33.3.3, "Transmit Internal/External Rate Modes." For Serial ATM based IMA PHYs the UPC must not be used.

43.4.2.2 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 32.3.2.5, "Global Mode Entry (GMODE)."

43.4.2.3 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 43-2. 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.

43.4.2.4 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.

43.4.3 IMA Root Table

Table 43-3. 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) 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 43.4.6.1, “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 43.4.3.2, “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 43.4.3.2, “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 43.4.3.2, “Utopia PHY Address Compression” for more details.

Table 43-3. IMA Root Table[1] (continued)

Offset	Name	Width	Description
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 43.4.3.2, "Utopia PHY Address Compression" for more details.
0x40–0x43	—	—	Reserved. Must be programmed to zero during initialization.
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 (i.e. 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.

43.4.3.1 IMA Control (IMACNTL)

The fields of the IMACNTL are shown in Figure 43-13.

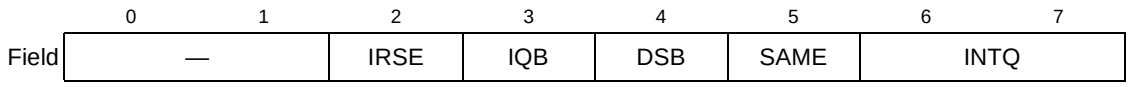


Figure 43-13. IMA Control (IMACNTL)

Table 43-4 describes the IMACNTL bit fields.

Table 43-4. 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.

43.4.3.2 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 43-14 depicts the compression scheme.

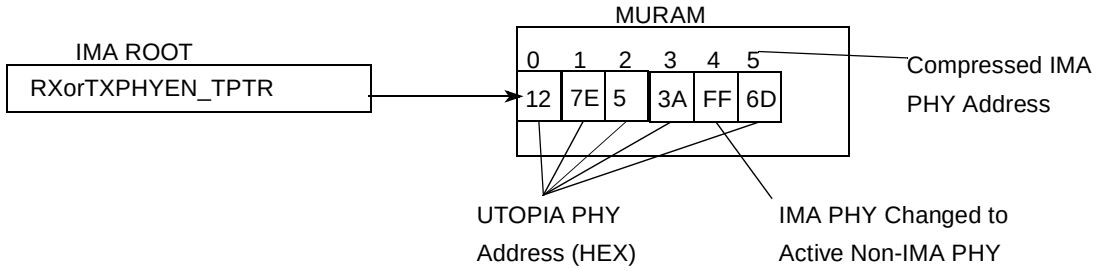


Figure 43-14. Utopia PHY Address Compression Tables

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 43-14 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.

43.4.4 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.

43.4.4.1 IMA Group Transmit Table Entry

Table 43-5. 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 (i.e. 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 (i.e. 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 43.4.4.1.1, “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 43-5. 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.

43.4.4.1.1 IMA Group Transmit Control (IGTCNTL)

The fields of the IGTCNTL register are shown in Figure 43-15.

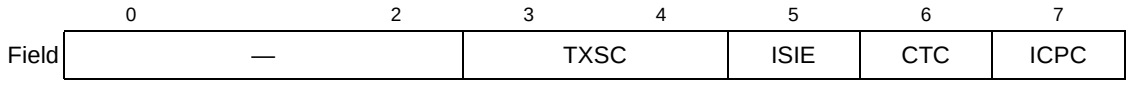


Figure 43-15. IMA Group Transmit Control (IGTCNTL)

Table 43-6 describes the IGTCNTL bit fields.

Table 43-6. 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.

43.4.4.1.2 IMA Group Transmit State (IGTSTATE)

The fields of the IGTSTATE register are shown in Figure 43-16.

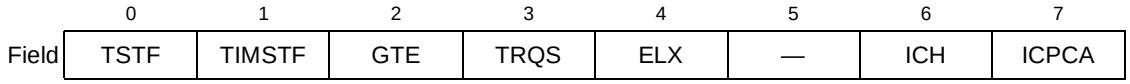


Figure 43-16. IMA Group Transmit State (IGTSTATE)

Table 43-7 describes the IGTSTATE bit fields.

Table 43-7. 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].

43.4.4.1.3 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 43-17.

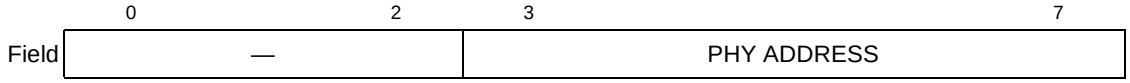


Figure 43-17. Transmit Group Order Table Entry

Table 43-8 describes the format of a transmit group order table entry.

Table 43-8. 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 43.4.3.2, "Utopia PHY Address Compression."

43.4.4.1.4 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 (i.e. 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 43-9 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 bitswapped. [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 43-9. 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 43-9. 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 43-9. 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.

43.4.4.2 IMA Group Receive Table Entry

Table 43-10. 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 (i.e., 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 43-10. 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 (i.e. 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 (i.e. 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 43-10. 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.

43.4.4.2.1 IMA Group Receive Control (IGRCNTL)

The fields of the IGRCNTL register are shown in Figure 43-18.

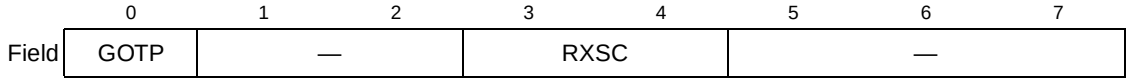


Figure 43-18. IMA Group Receive Control (IGRCNTL)

Table 43-11 describes the IGRCNTL bit fields.

Table 43-11. 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, i.e., 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.

43.4.4.2.2 IMA Group Receive State (IGRSTATE)

The fields of the IGRSTATE register are shown in Figure 43-19.

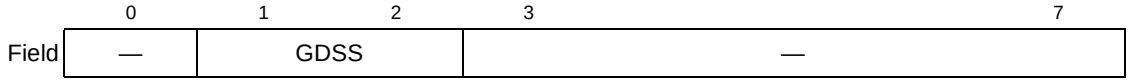


Figure 43-19. IMA Group Receive State (IGRSTATE)

Table 43-12 describes the IGRSTATE bit fields.

Table 43-12. 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 43.5.3.10, “Receive Event Response Procedures.”
3–7	—	Reserved, initialize to zero.

43.4.4.2.3 IMA Receive Group Frame Size

The fields of the IRGFS register are shown in Figure 43-20.

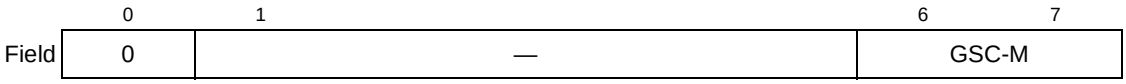


Figure 43-20. IMA Receive Group Frame Size (IGRSTATE)

Table 43-13 describes the IRGFS bit fields.

Table 43-13. 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

43.4.4.2.4 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 (i.e. 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 43-21.

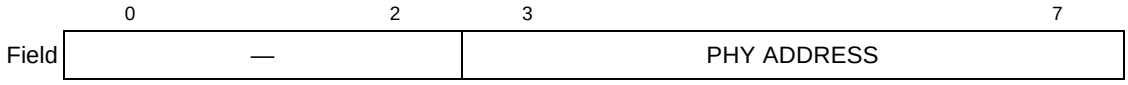


Figure 43-21. Receive Group Order Table Entry

Table 43-14 describes the format of a receive group order table entry.

Table 43-14. 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 43.4.3.2, “Utopia PHY Address Compression.”

43.4.5 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.

43.4.5.1 IMA Link Transmit Table Entry

Table 43-15. 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 43-15. 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 43.4.6.1, “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 43.4.6.1, “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 43.4.6.1, “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 43.4.6.1, “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.

43.4.5.1.1 IMA Link Transmit Control (ILTCNTL)

The fields of the ILTCNTL register are shown in Figure 43-22.

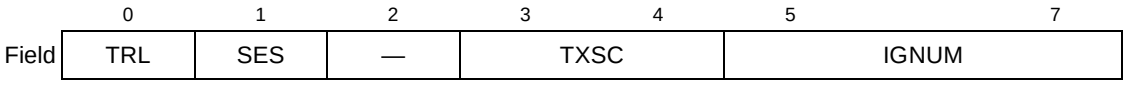


Figure 43-22. IMA Link Transmit Control (ILTCNTL)

Table 43-16 describes the ILTCNTL bit fields.

Table 43-16. 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.

43.4.5.1.2 IMA Link Transmit State (ILTSTATE)

The fields of the ILTSTATE register are shown in Figure 43-23

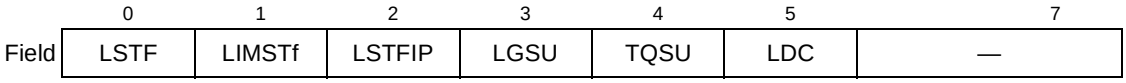


Figure 43-23. IMA Link Transmit State (ILTSTATE)

Table 43-17 describes the ILTSTATE bit fields.

Table 43-17. 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 43-17. 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.

43.4.5.1.3 IMA Transmit Interrupt Status (ITINTSTAT)

The fields of the ITINTSTAT register are shown in Figure 43-24.

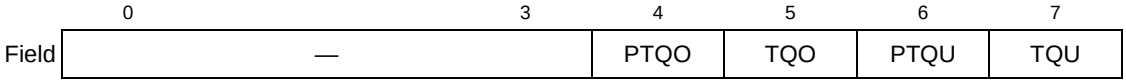


Figure 43-24. IMA Transmit Interrupt Status (ITINTSTAT)

Table 43-18 describes the ITINTSTAT bit fields.

Table 43-18. 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.

43.4.5.2 IMA Link Receive Table Entry

Table 43-19. 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 43.4.6.2, “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 43.4.6.2, “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 43-19. 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.

43.4.5.2.1 IMA Link Receive Control (ILRCNTL)

The fields of the ILRCNTL register are shown in Figure 43-25

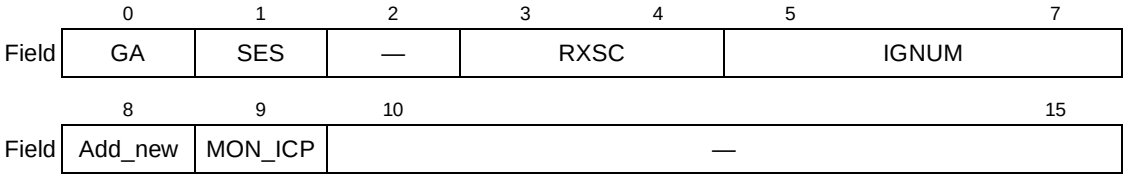


Figure 43-25. IMA Link Receive Control (ILRCNTL)

Table 43-20 describes the ILRCNTL bit fields.

Table 43-20. 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 43-20. 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 (i.e. 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.

43.4.5.2.2 IMA Link Receive State (ILRSTATE)

The fields of the ILRSTATE register are shown in Figure 43-26

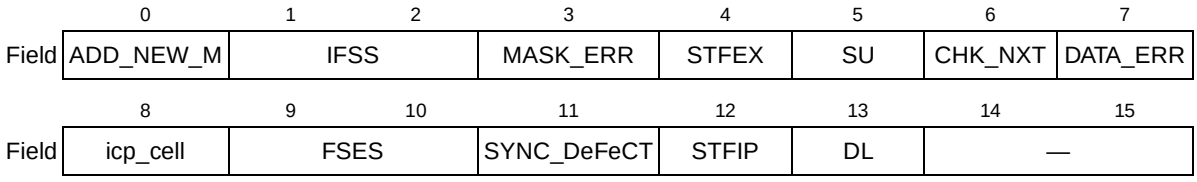


Figure 43-26. IMA Link Receive State (ILRSTATE)

Table 43-21 describes the ILRSTATE bit fields.

Table 43-21. 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 43-21. 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.

43.4.5.3 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 43-22

Table 43-22. 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.

43.4.6 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].

43.4.6.1 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.

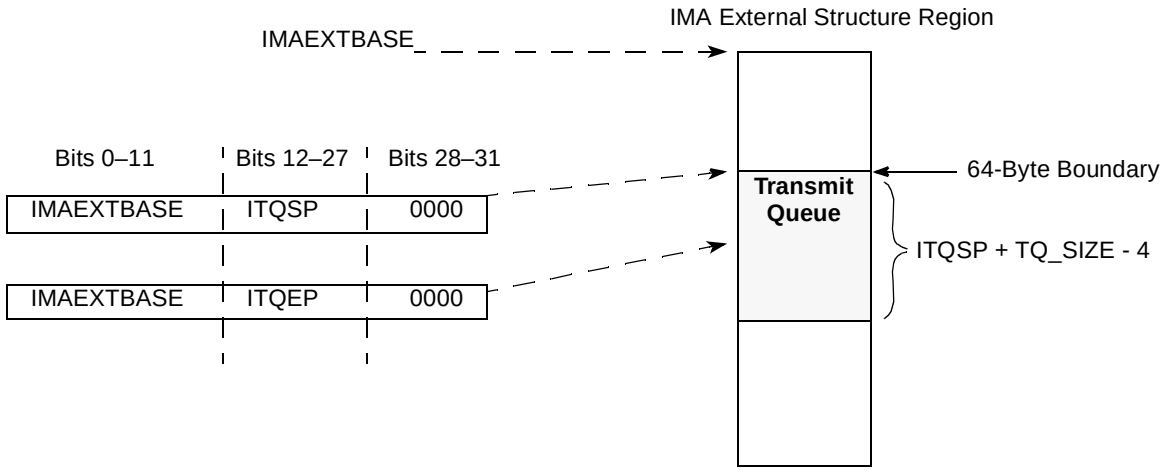


Figure 43-27. IMA Transmit Queue

43.4.6.2 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.

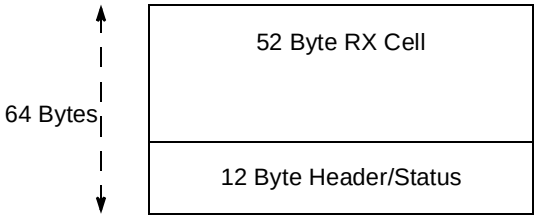


Figure 43-28. Cell Buffer in Delay Compensation Buffer

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.

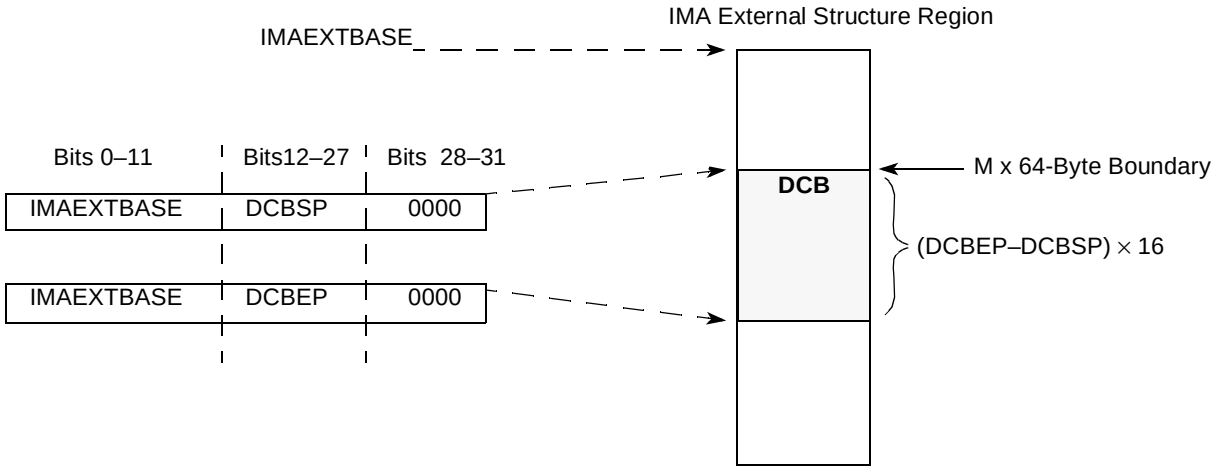


Figure 43-29. IMA Delay Compensation Buffer

43.4.7 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.

43.4.7.1 IMA Interrupt Queue Entry

The format for the IMA interrupt queue entries is shown in Figure 43-30

	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 43-30. IMA Interrupt Queue Entry

Table 43-23 describes the IMA interrupt queue entry bit fields.

Table 43-23. 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 (i.e. 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.

43.4.7.2 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.

43.4.8 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 43-24.

Table 43-24. 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.

43.4.8.1 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.

43.4.9 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 43-31.
2. To issue this UCC command, refer to Section 20.3.1.1, “QUICC Engine Commands.” Use opcode 001101(0x0D).

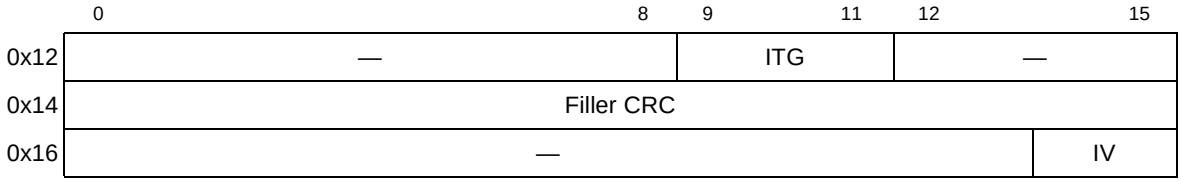


Figure 43-31. COMM_INFO Field

Table 43-25. 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 20.3.1.1, “QUICC Engine Commands.”

43.5 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.

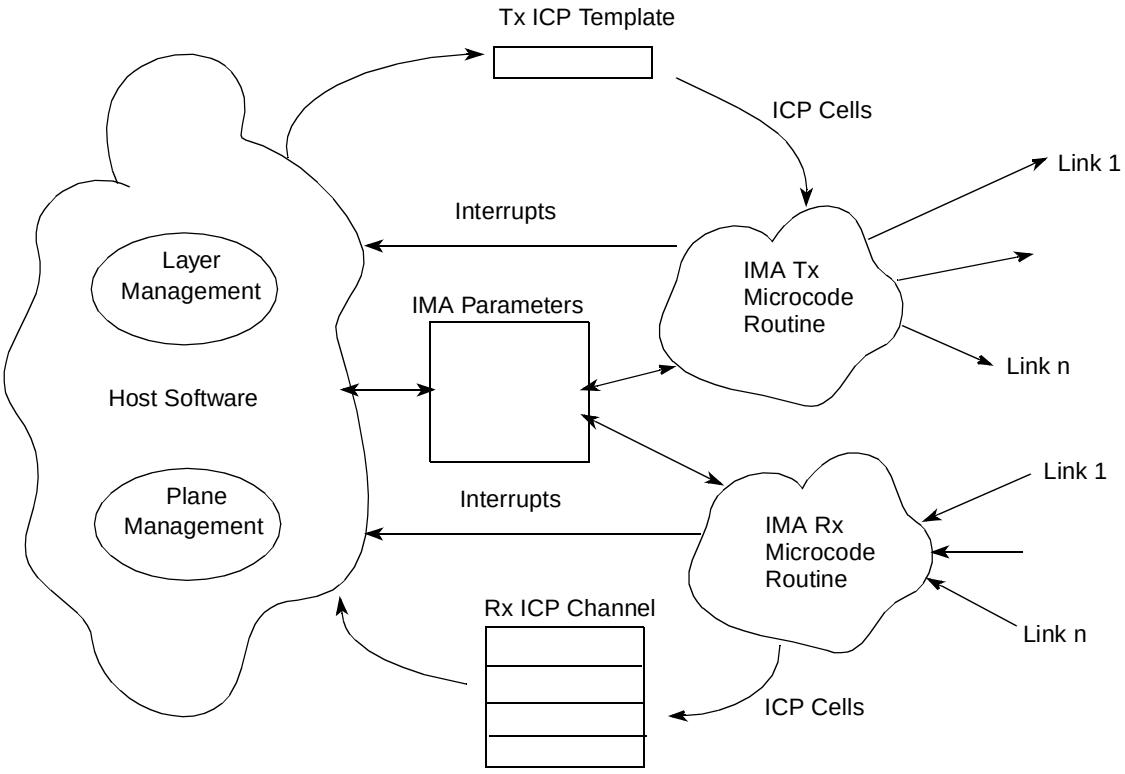


Figure 43-32. IMA Microcode/Software Interaction

43.5.1 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 (e.g. host commands, RCT/TCT parameters, buffer descriptors, interrupts).

43.5.2 Software Responsibilities

Table 43-26 lists functions that are the responsibility of the host software, which must complement the IMA microcode to provide a complete IMA solution.

Table 43-26. 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 (e.g. 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 43-26. 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
Performance parameter measurement and reporting	Establish timers to correlate errors with time intervals (e.g. to determine severely errored seconds (SES) or unusable seconds (UUS))
	Maintain statistics.

Table 43-26. Host Software Functions (continued)

Function	Action
	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.

43.5.3 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.

43.5.3.1 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].

43.5.3.2 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.

43.5.3.3 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.

43.5.3.4 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.

43.5.3.4.1 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 43.5.3.1, “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 (i.e. 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.

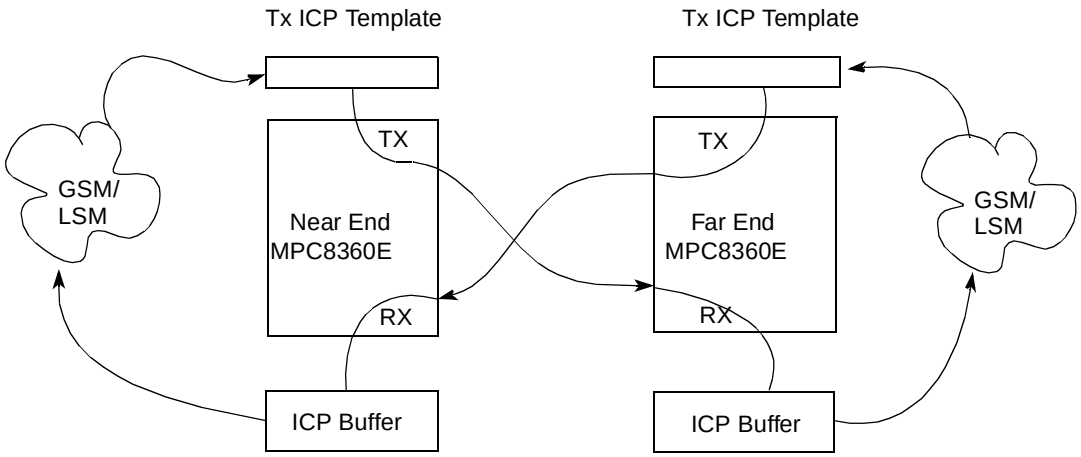


Figure 43-33. Near-End versus Far-End

43.5.3.4.2 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 32, "ATM Controller AAL0, AAL1, and AAL5."

43.5.3.5 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 43.4.4.2.4, “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 43.4.4.2, “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 43.5.3.4, “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 43.5.3.4, “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 43.4.4.2.4, “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 43.4.4.2, “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 43-3.
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 43.4.4.1.4, “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.

43.5.3.6 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.

43.5.3.6.1 Rx Link Removal

1. Formulate new group order table with the “dropped” link excluded (see Section 43.4.4.2.4, “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 (i.e., 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 43.5.3.6.2, “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 43.5.3.6.2, “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.

43.5.3.6.2 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 (e.g. no RxClav assertions for all links in the group), the removal procedure in Section 43.5.3.6.1, “Rx Link Removal” will not be successful. Therefore, if all links in a group are no longer passing cells, follow the procedure in Section 43.5.3.6.2, “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 43.5.3.4, “Group Startup Procedure”).

43.5.3.6.3 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 43.4.4.1.3, “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).

43.5.3.6.4 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.

43.5.3.7 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 43.4.4.2, “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 (i.e., 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 43.5.3.6.2, “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 43.4.4.2.4, “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 (i.e. involved in passing cells from the DCB to the ATM microcode) software must also clear IGRSTATE[GDSS].

43.5.3.8 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 43.4.4.2.4, “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 43.4.4.2, “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 43-3.
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.

43.5.3.9 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 43.5.3.6, “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 43.5.3.6, “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.

43.5.3.10 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 43.5.3.6, “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 43.5.3.6, “Link Removal Procedure” or Section 43.5.3.7, “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 43.5.3.5, “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 43.5.3.5, “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 43.4.5.3, “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 43.5.3.6, “Link Removal Procedure or Section 43.5.3.7, “Link Receive Deactivation Procedure. The link can then be re-added to the group using the link addition procedure see Section 43.5.3.5, “Link Addition Procedure for a removed link or Section 43.5.3.8, “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 43.5.3.7, “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 43.5.3.8, “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 43-34. 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.

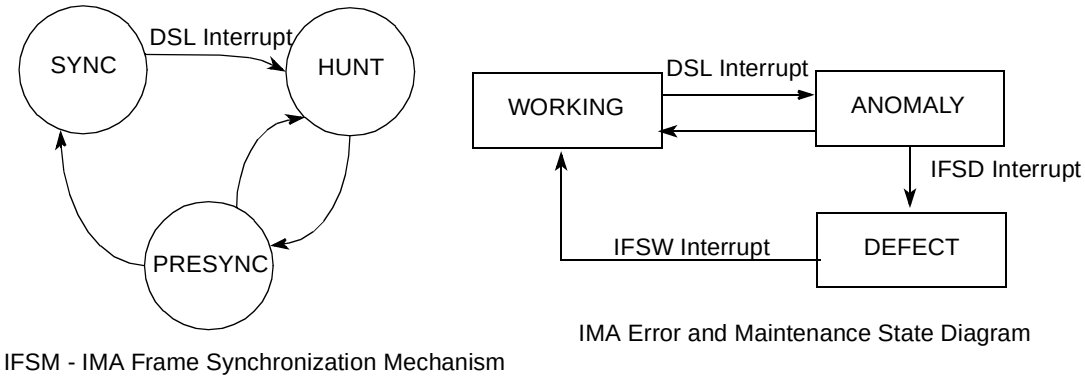


Figure 43-34. Receive Event Generation For ATM Forum IMA State Machines

43.5.3.11 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.

43.5.3.11.1 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 43.5.3.1, “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).

43.5.3.11.2 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 43.5.3.1, “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.

43.5.3.12 End-to-End Channel Signaling Procedure

43.5.3.12.1 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.

43.5.3.12.2 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 44
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.

44.1 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).

44.1.1 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). Note that low-speed operation requires an external hub.
- 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)

44.2 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 44.4.1.1, “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.

44.2.1 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.

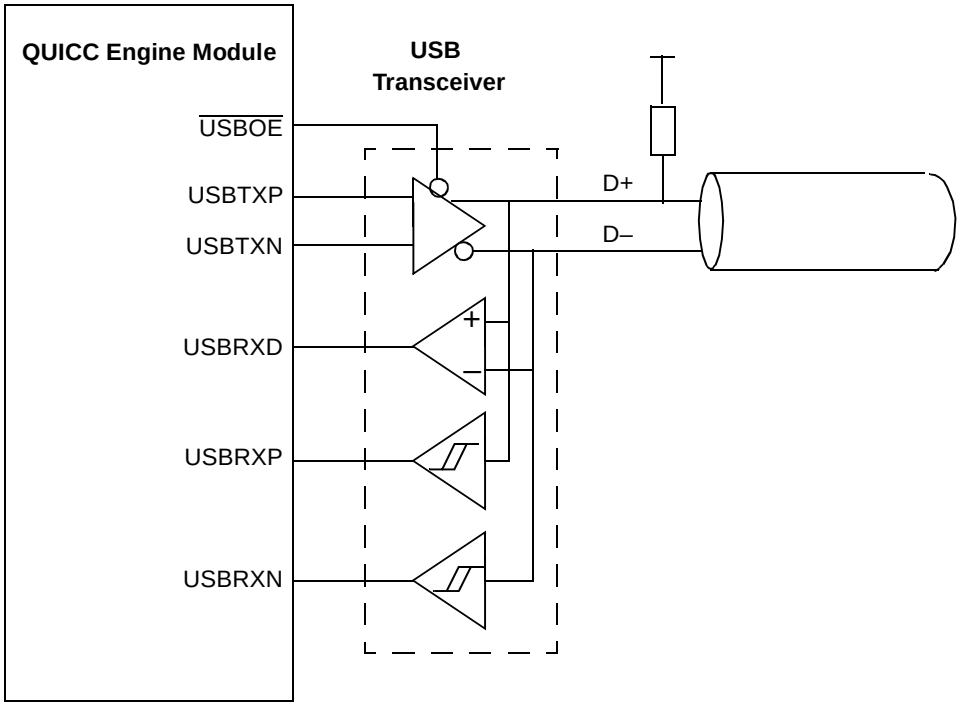


Figure 44-1. USB Interface

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 21.5.1, "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 44-1. 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 44-1. USB Pin Functions

<table>
<tr><th>Signal</th><th>I/O</th><th>Function</th></tr>
<tr><td>USBTXN,
USBTXP</td><td>O</td><td>Outputs from the USB transmitter, inputs to the differential driver.
<table>
<tr><th>TP</th><th>TN</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>Single-ended 0</td></tr>
<tr><td>0</td><td>1</td><td>Logic 0</td></tr>
<tr><td>1</td><td>0</td><td>Logic 1</td></tr>
<tr><td>1</td><td>1</td><td>—</td></tr>
</table></td></tr>
<tr><td>$\overline{\text{USBOE}}$</td><td>O</td><td>Output enable. Enables the transceiver to send data on the bus.</td></tr>
</table>

Table 44-1. 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>

44.3 USB Function Description

As shown in Figure 44-2, 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.

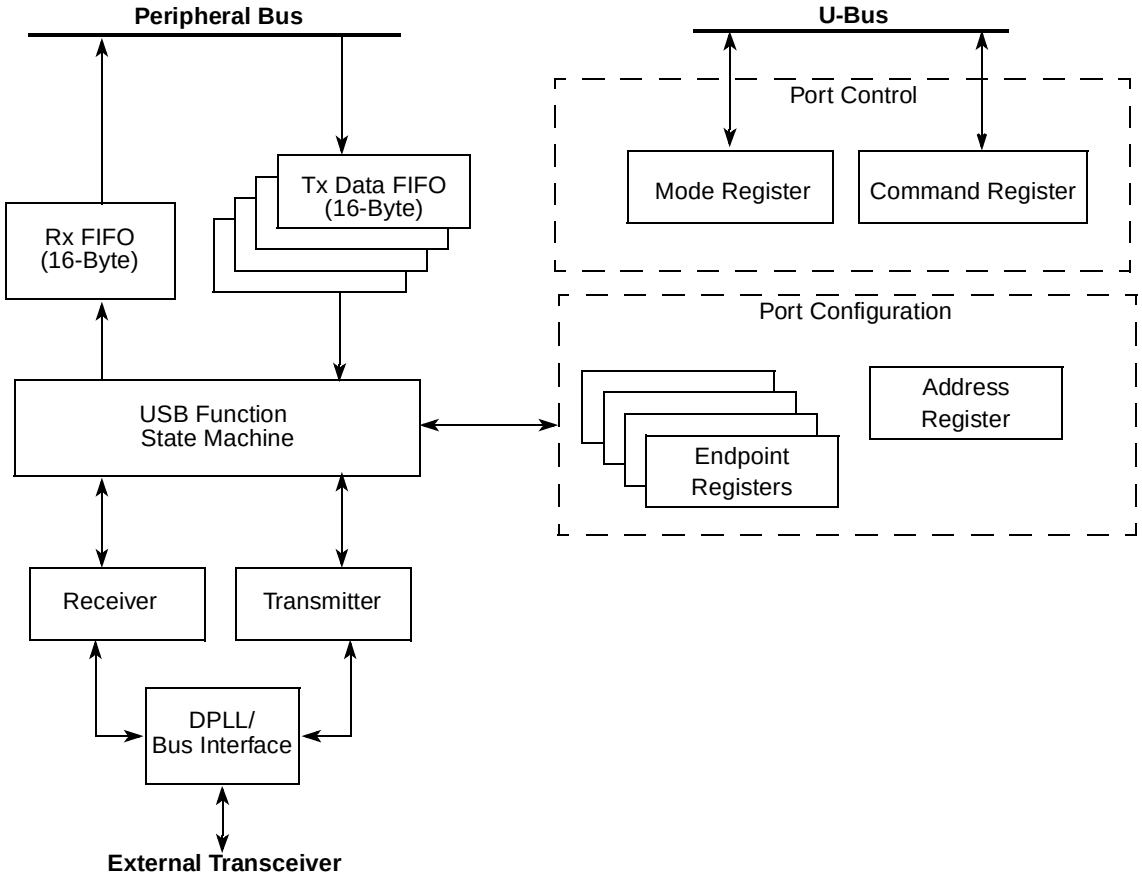


Figure 44-2. USB Function Block Diagram

44.3.1 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 44.4.7.2, “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 44.4.7.3, “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 44-3 and Table 44-2 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).

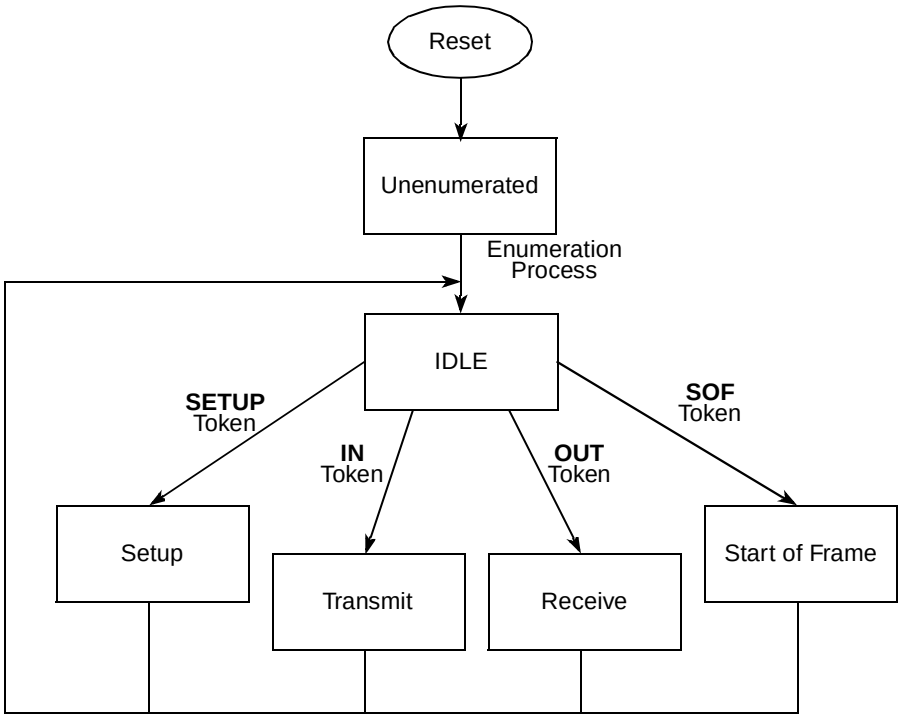


Figure 44-3. USB Controller Operating Modes

Table 44-2. 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 44-2. 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.

44.4 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 44-4). 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.

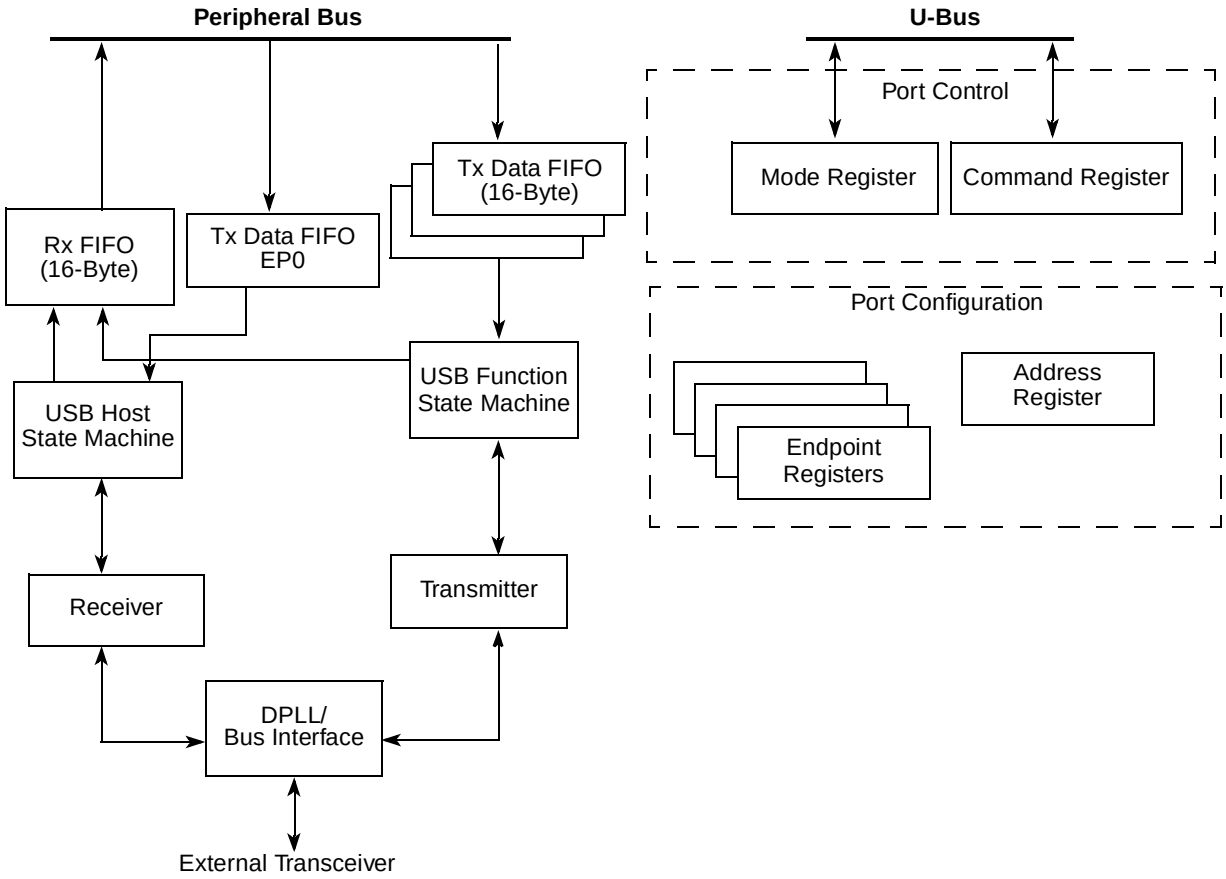


Figure 44-4. USB Controller Block Diagram

44.4.1 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 44.4.7.1, “USB Mode Register (USMOD)”). USEP1 should be programmed for host operation as described in Section 44.4.7.3, “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 44-3 and Table 44-2 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 44.4.7, “USB Function Programming Model.”

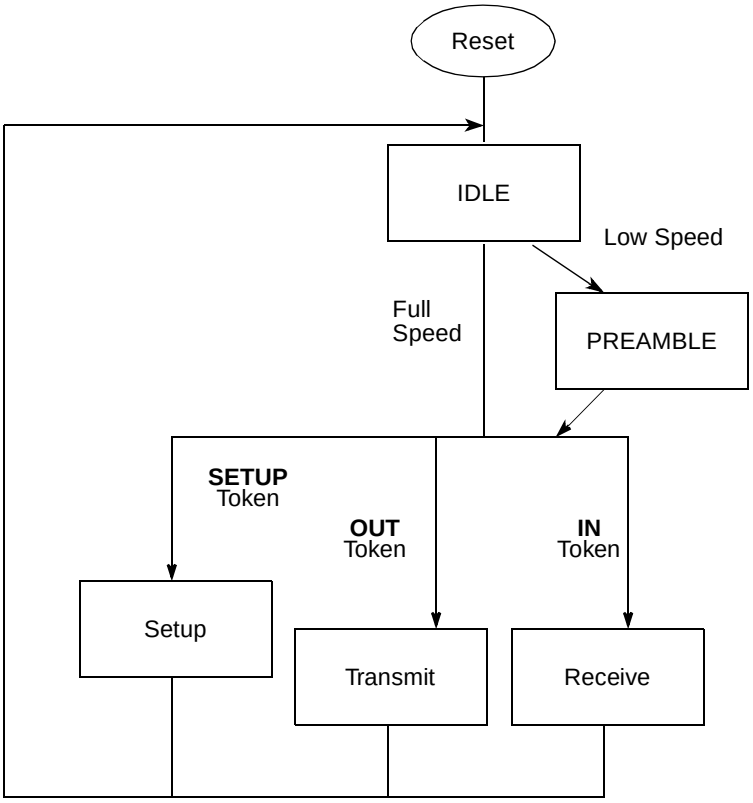


Figure 44-5. USB Controller Operating Modes

44.4.1.1 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 44.5.3, "USB Transmit Buffer Descriptor (TxBD) for Host." Each receive packet is stored in a buffer referenced by a RxBD as described in Section 44.5.1, "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.

44.4.1.2 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 44.5.4, "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 44-3. 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 44-3. 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 non-octet sized data and RxBD[AB] if bit stuffing error occurred.
If no valid DATA PID or no data at all received during the expected time interval a TO indication in the token TxBD is set.</td><td>Transaction-Level Interface

Transmission begins when the USB host controller fetches a TrBD with the TOK field indicating an IN transaction. After the IN token is generated and transmitted, the USB host controller waits for reception of data within the expected time interval. The received data packet is stored in buffer reference by the TrBD. While receiving the data the USB host controller calculates CRC16 and performs bit un-stuffing. At end of the packet, the calculated CRC is compared to the received value and octet alignment is checked, TrBD[R] is cleared, TrBD[PID] is set according to the received DATA PID and error indications are set if required: TrBD[CR] for failed CRC check, TrBD[NO] for non-octet sized data and TrBD[AB] if bit stuffing error occurred. If any of the above errors are reported, TrBD[RXER] is also set, and a TXE1 interrupt is generated. If no valid DATA PID or no data at all received during the expected time interval, a TrBD[TO] is set and a TXE1 interrupt is generated. If no errors occurred and TrBD[I] is set, a TXB interrupt is generated to indicate successful completion of the transaction.</td></tr>
<tr><td 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>

44.4.2 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 44.4.7.8, "USB Start of Frame Timer (USSFT)" and Section 44.4.7.9, "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.

44.4.3 USB Function and Host Parameter RAM Memory Map

The USB controller parameter RAM area, shown in Table 44-4, 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 44-4. 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 44-6. The map of the endpoint parameter block is shown in Table 44-5 **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.

44.4.4 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 44-6.

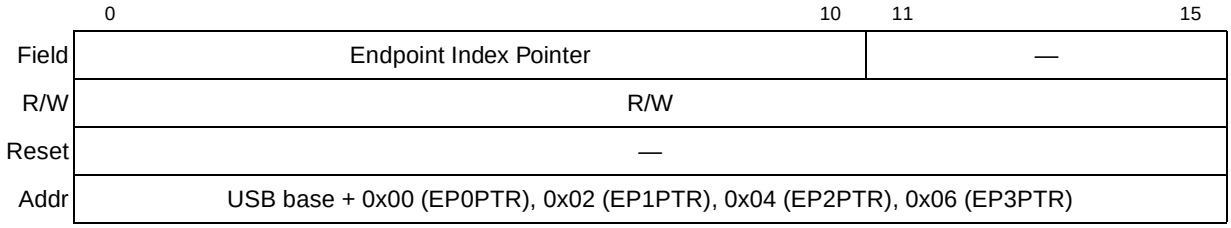


Figure 44-6. Endpoint Pointer Registers (EP*n*PTR)

The map of the endpoint parameter block is shown in Table 44-5.

Table 44-5. 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 44.4.6, "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 44-5. 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	

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.

44.4.5 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.

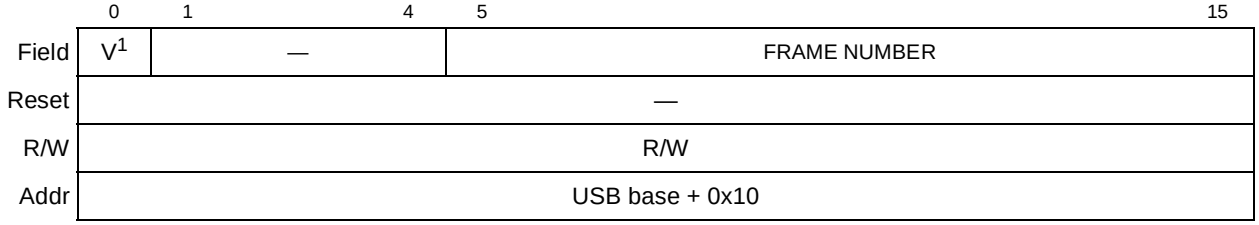


Figure 44-7. Frame Number (FRAME_N) in Function Mode—Updated by USB Controller

[1] This bit is set if the SOF token is received error free.

Table 44-6 describes FRAME_N fields in function mode.

Table 44-6. 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 44.4.1.2, “Transaction-Level Interface,” for details. The software should prepare the frame number and the CRC and place it in FRAME_N field.

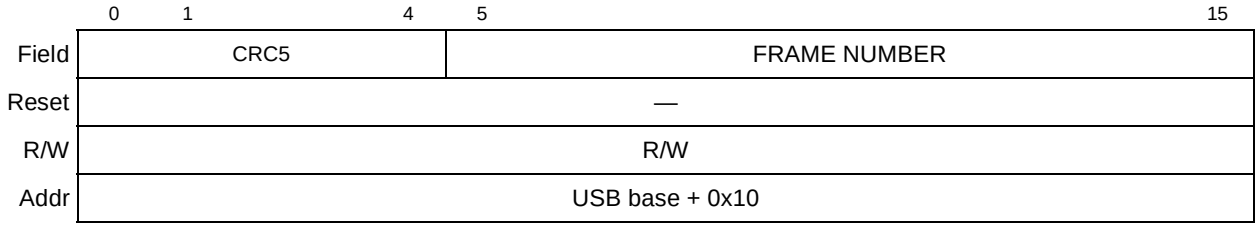


Figure 44-8. Frame Number (FRAME_N) in Host Mode—Updated by Application Software

Table 44-7 describes FRAME_N fields in host mode.

Table 44-7. 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.

44.4.6 USB Bus Mode Registers (RBMR and TBMR)

Figure 44-9 shows the fields in the receive/transmit bus mode registers.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		CETM	DTB	—

Figure 44-9. USB Bus Mode Registers (RBMR and TBMR)

Table 44-8 describes RBMR and TBMR fields.

Table 44-8. 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 19.1.8.2, “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.

44.4.7 USB Function Programming Model

The following sections describe USB controller registers.

44.4.7.1 USB Mode Register (USMOD)

USMOD, shown in Figure 44-10, 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 44-10. USB Mode Register (USMOD)

Table 44-9 describes USMOD fields.

Table 44-9. 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 44.4.7.8, “USB Start of Frame Timer (USSFT)” and Section 44.4.2, “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.

44.4.7.2 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 44-11. USB Slave Address Register (USADD)

Table 44-10 describes the USADR fields.

Table 44-10. 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.

44.4.7.3 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 44-12. USB Endpoint Registers (USEP0–USEP3)

Table 44-11 describes the fields of USEP0–USEP3. The setting for USB host controller should be set only in USEP0, when USMOD[HOST] is set.

Table 44-11. 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 44.4.1.1, “Packet-Level Interface.” 1 Transaction-level interface as described in Section 44.4.1.2, “Transaction-Level Interface.”

Table 44-11. 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

44.4.7.4 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 44-13. USB Command Register (USCOM)

Table 44-12 describes the USCOM fields.

Table 44-12. 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 44.6, “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 44-12. 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.

44.4.7.5 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 44-14. USB Event Register (USBER)

Table 44-13 describes USBER fields.

Table 44-13. 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.

44.4.7.6 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.

44.4.7.7 USB Status Register (USBS)

USBS, described in Figure 44-15 and Table 44-14, allows the user to monitor real-time status condition on the USB lines.

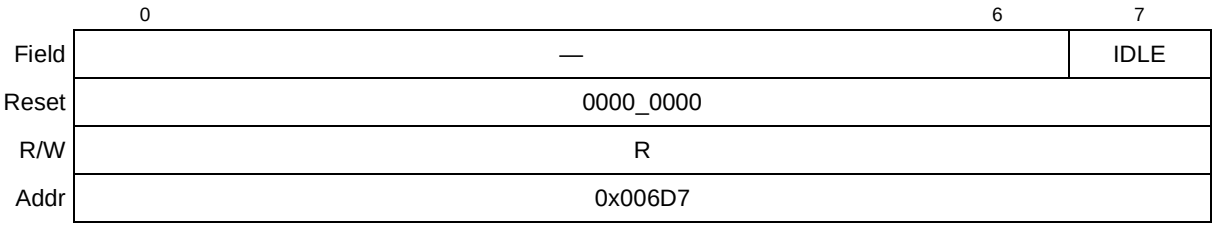


Figure 44-15. USB Status Register (USBS)

Table 44-14 describes USBS fields.

Table 44-14. 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.

44.4.7.8 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.

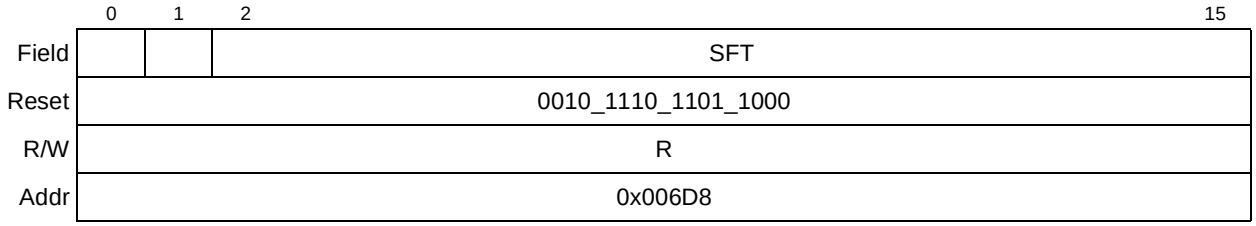


Figure 44-16. USB Start of Frame Timer (USSFT)

Table 44-15 describes the USSFT fields.

Table 44-15. 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.

44.4.7.9 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.

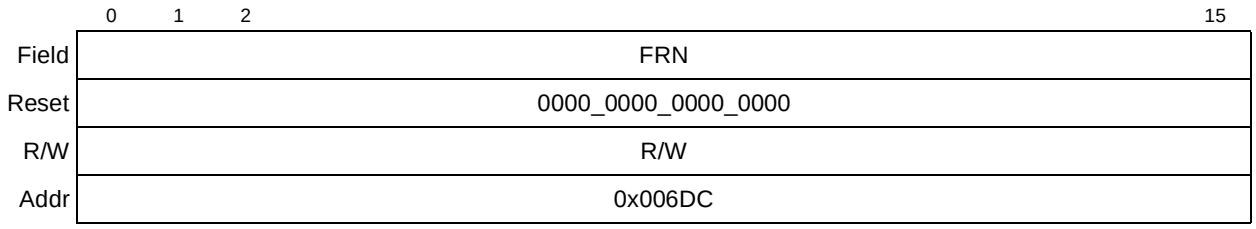


Figure 44-17. USB Frame Number (USFRN)

Table 44-16 describes the USFRN fields.

Table 44-16. 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.

44.5 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 44-18). 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.

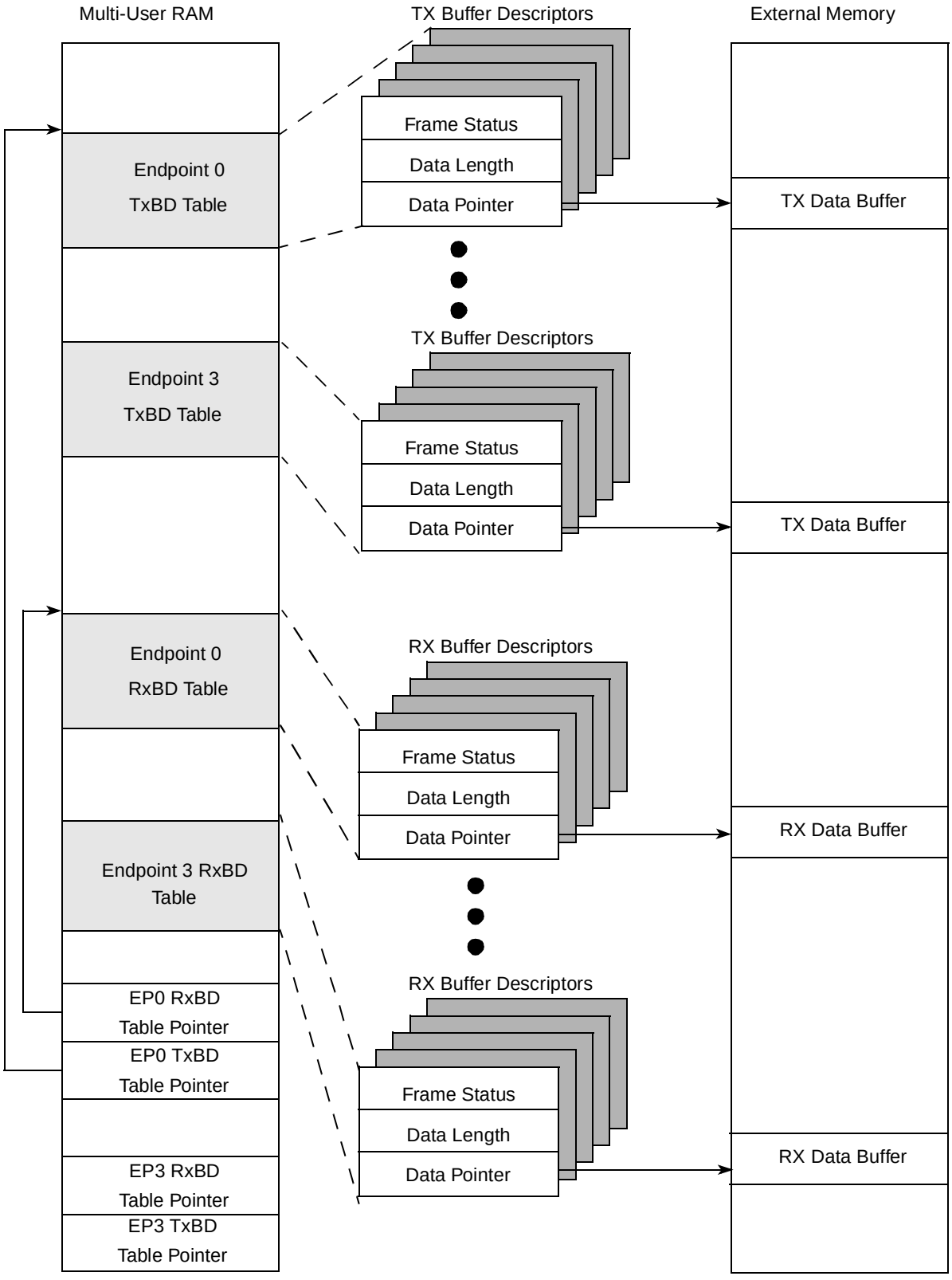


Figure 44-18. USB Memory Structure

44.5.1 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 44-19, 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.

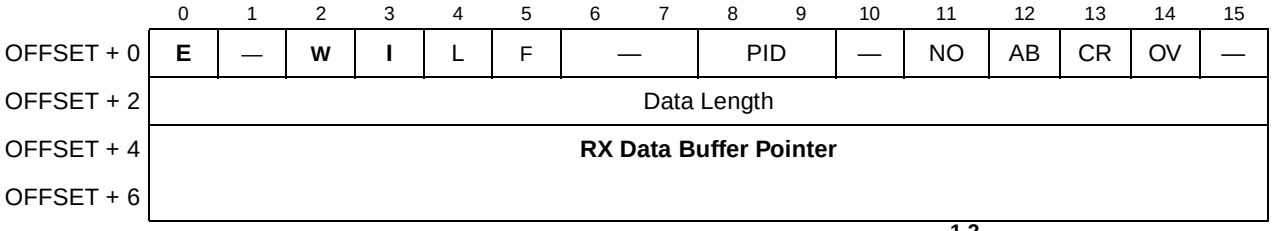


Figure 44-19. USB Receive Buffer Descriptor (RxBD)[1,2]

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 44-17 describes USB receive buffer descriptor fields.

Table 44-17. 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 44-17. USB RxBD Fields (continued)

Offset	Bits	Name	Description
	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 non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data 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.

44.5.2 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.

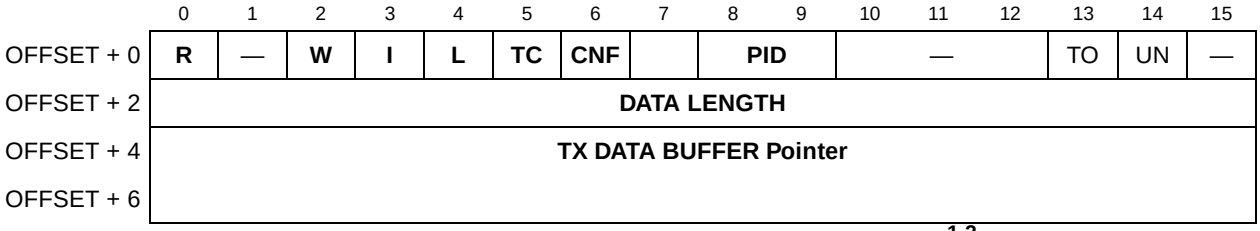


Figure 44-20. USB Transmit Buffer Descriptor (TxBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 44-18 describes USB TxBD fields.

Table 44-18. 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 44-18. USB Function TxBD Fields (continued)

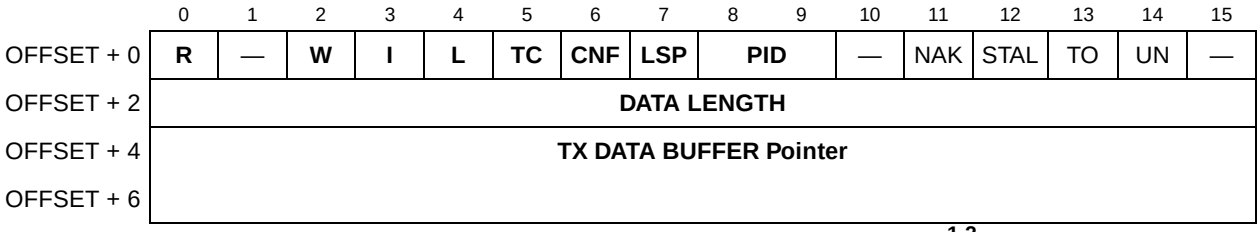
Offset	Bits	Name	Description
	6	CNF	Transmit confirmation. Valid only when the L bit is set; otherwise it is ignored. Applies to multi-frame enabled endpoints (USEP*n*[MF] = 1); refer to Section 44.4.7.3, "USB Endpoint Registers (USEP0–USEP3)." 0 Continue to load the transmit FIFO with the next packet. 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.

44.5.3 USB Transmit Buffer Descriptor (TxBD) for Host

The TxBD described in this section is used when the packet-level interface is active. See Section 44.4.1.1, "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 44-21. 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 44-18 describes USB TxBD fields.

Table 44-19. 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 44-19. USB Host TxBD Fields (continued)

Offset	Bits	Name	Description
	6	CNF	Transmit confirmation. Valid only when the L bit is set; otherwise it is ignored. Applies to multi-frame enabled endpoints (USEP*n*[MF] = 1); see Section 44.4.7.3, "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.

44.5.4 USB Transaction Buffer Descriptor (TrBD) for Host

The TrBD described in this section is used when the transaction-level interface is active. See Section 44.4.1.2, "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.

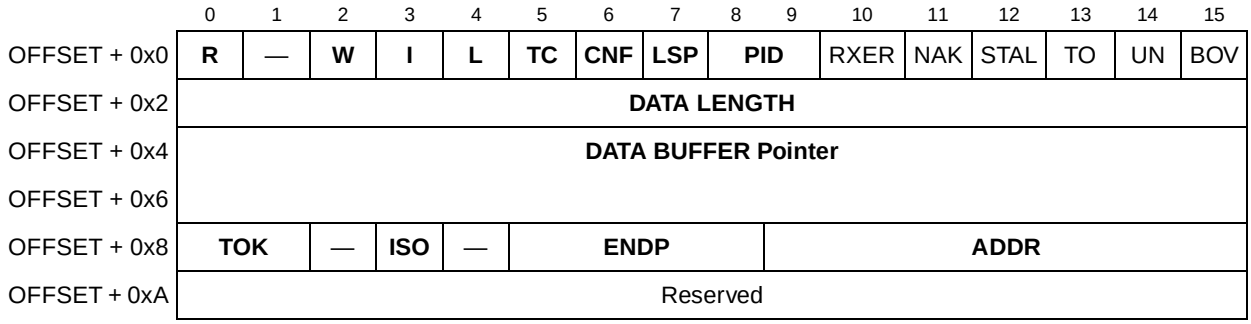


Figure 44-22. USB Transaction Buffer Descriptor (TrBD)[1,2]

1 Entries in **boldface** must be initialized by the user.

2 All fields should be prepared by the user before transmission.

Table 44-18 describes USB TrBD fields.

Table 44-20. 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 44-20. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
	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 non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	STAL/AB[1]	RXER = 0: STALL received. Indicates that the endpoint has responded with a STALL handshake (OUT transaction). The endpoint needs attention through the control pipe. RXER = 1: Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	13	TO/CR[1]	RXER = 0: 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 44-20. USB Host TrBD Fields (continued)

Offset	Bits	Name	Description
	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.

[1] Written by the USB controller after it finishes sending or receiving the associated data buffer.

44.6 USB QUICC Engine Commands

The following transmit commands are issued to the QUICC Engine command register (CECR). Refer to Section 20.4.1, “QUICC Engine Command Register (CECR).”

44.6.1 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.

44.6.2 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).

44.7 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 44-21. Errors that exist exclusively in host mode or function mode are marked as such.

Table 44-21. 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 44-22 describes the USB controller reception errors.

Table 44-22. 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
MPC8358E

This appendix illustrates the MPC8358E and in particular the differences between it and the MPC8360E as described in this manual.

A.1 Features

Major features of the MPC8358E are as follows:

- e300c1 PowerPC processor core
  - Enhanced version of the MPC603e processor core
  - High-performance, superscalar processor core
  - Floating-point, integer, load/store, system register, and branch processing units
  - 32-Kbyte instruction cache, 32-Kbyte data cache
  - Dynamic power management
  - Enhanced hardware program debug features
  - Software-compatible with the Freescale processor families implementing the PowerPC architecture
- QUICC Engine unit
  - Two 32-bit RISC controllers for flexible support of the communications peripherals
  - Serial DMA channel for receive and transmit on all serial channels
  - QE peripheral request interface (for SEC, PCI, IEEE Std 1588™)
  - Six UCCs supporting the following protocols and interfaces (not all of them simultaneously):
    - 10/100 Mbps Ethernet/IEEE Std 802.3™ through MII and RMII interfaces.[1]
    - 1000 Mbps Ethernet/IEEE Std 802.3 through a media-independent interface (GMII, RGMII, RTBI, TBI) on UCC1 and UCC2
    - 10/100 Mbps Ethernet/IEEE Std 802.3 L2 switch port through MII and RMII interfaces.
    - IEEE Std 1588 protocol supported
    - 9.6K jumbo frames
    - ATM full-duplex SAR, up to 622 Mbps (OC-12/STM-4), AAL0, AAL1 and AAL5 in accordance ITU-T I.363.5
    - ATM AAL2 CPS, SSSAR, and SSTED up to 155 Mbps (OC-3/STM-1) Mbps full duplex (with 4 CPS packets per cell) in accordance ITU-T I.366.1 and I.363.2
    - ATM traffic shaping for CBR, VBR, UBR, and GFR traffic types compatible with ATM forum TM4.1 for up to 64K simultaneous ATM channels
    - ATM AAL1 structured and unstructured circuit emulation service (CES 2.0) in accordance with ITU-T I.163.1 and ATM Forum af-vtoa-00-0078.000

1.SMII or SGMII media-independent interface is not currently supported

- IMA (Inverse Multiplexing over ATM) for up to 31 IMA links over 8 IMA groups in accordance with the ATM forum AF-PHY-0086.000 (Version 1.0) and AF-PHY-0086.001 (Version 1.1)
- ATM Transmission Convergence layer support in accordance with ITU-T I.432
- ATM OAM handling features compatible with ITU-T I.610
- PPP, Multi-Link (ML-PPP), Multi-Class (MC-PPP) and PPP mux in accordance with the following RFCs: 1661, 1662, 1990, 2686 and 3153
- IP support for IPv4 and IPv6 packets including TOS, TTL and header checksum processing
- Ethernet over first mile IEEE Std 802.3ah™
- Shim header
- Ethernet-to-Ethernet/AAL5/AAL2 inter-working
- L2 Ethernet switching using MAC address or IEEE Std 802.1P/Q™ VLAN tags
- ATM (AAL2/AAL5) to Ethernet (IP) interworking in accordance with RFC2684 including bridging of ATM ports to Ethernet ports
- Extensive support for ATM statistics and Ethernet RMON/MIB statistics
- AAL2 protocol rate up to 4 CPS at OC-3/STM-1 rate
- Packet over Sonet (POS) up to 622-Mbps full-duplex 124 MultiPHY[1]
- POS hardware; microcode must be loaded as an IRAM package
- Transparent up to 70-Mbps full-duplex
- HDLC up to 70-Mbps full-duplex
- HDLC BUS up to 10 Mbps
- Asynchronous HDLC
- UART
- BISYNC up to 2 Mbps
- User-programmable FIFO size
- QUICC Multichannel Controller (QMC) for 64 TDM channels

— One UTOPIA/POS interface supporting 31/124 MultiPHY

— Universal serial bus (USB) controller

— Two serial peripheral interfaces (SPI); SPI2 is dedicated to Ethernet PHY management

— Four TDM interfaces with 1-bit mode for E3/T3 rates in clear channel

— Sixteen independent baud rate generators and 30 input clock pins for supplying clocks to UCC serial channels

— Four independent 16-bit timers that can be interconnected as four 32-bit timers

— Interworking functionality:

- Layer 2 10/100-Base T Ethernet switch
- ATM-to-ATM switching (AAL0, 2, 5)
- Ethernet-to-ATM switching with L3/L4 support
- PPP interworking

- Security engine is optimized to handle all the algorithms associated with IPSec, SSL/TLS, SRTP, 802.11i, iSCSI, and IKE processing. The security engine contains four crypto-channels, a controller, and a set of crypto execution units (EUs).
  - Public key execution unit (PKEU) supporting the following:
    - RSA and Diffie-Hellman
    - Programmable field size up to 2048 bits
    - Elliptic curve cryptography
    - F2m and F(p) modes
    - Programmable field size up to 511 bits
  - Data encryption standard execution unit (DEU)
    - DES, 3DES
    - Two key (K1, K2) or three key (K1, K2, K3)
    - ECB and CBC modes for both DES and 3DES
  - Advanced encryption standard unit (AESU)
  - Implements the Rinjdael symmetric key cipher
  - Key lengths of 128, 192, and 256 bits, two key
    - ECB, CBC, CCM, and counter modes
  - ARC four execution unit (AFEU)
    - Implements a stream cipher compatible with the RC4 algorithm
    - 40- to 128-bit programmable key
  - Message digest execution unit (MDEU)
    - SHA with 160-, 224-, or 256-bit message digest
    - MD5 with 128-bit message digest
    - HMAC with either SHA or MD5 algorithm
  - Random number generator (RNG)
  - Four crypto-channels, each supporting multi-command descriptor chains
    - Static and/or dynamic assignment of crypto-execution units via an integrated controller
    - Buffer size of 256 bytes for each execution unit, with flow control for large data sizes
  - Storage/NAS XOR parity generation accelerator for RAID applications
- DDR SDRAM memory controller
  - Programmable timing supporting DDR SDRAM
  - DDR bus can be configured as a 32-bit or 64-bit bus
  - 32- or 64-bit data interface
  - Four banks of memory, each up to 1 Gbyte
  - DRAM chip configurations from 64 Mbits to 1 Gigabit with x8/x16 data ports
  - Full ECC support
  - Page mode support (up to 16 simultaneous open pages for DDR1, up to 32 simultaneous open pages for DDR2)

- Contiguous or discontiguous memory mapping
- Read-modify-write support
- Sleep mode support for self refresh SDRAM
- Supports auto refreshing
- Supports source clock mode
- On-the-fly power management using CKE
- Registered DIMM support
- 2.5-V SSTL2 compatible I/O for DDR1, 1.8-V SSTL2 compatible I/O for DDR2
- External driver impedance calibration
- On-die termination (ODT)

- PCI interface
  - PCI Specification Revision 2.3 compatible
  - Data bus widths:
    - Single 32-bit data PCI interface that operates at up to 66 MHz
  - PCI 3.3-V compatible (not 5-V compatible)
  - PCI host bridge capabilities on both interfaces
  - PCI agent mode supported on PCI interface
  - Support for PCI-to-memory and memory-to-PCI streaming
  - Memory prefetching of PCI read accesses and support for delayed read transactions
  - Support for posting of processor-to-PCI and PCI-to-memory writes
  - On-chip arbitration, supporting five masters on PCI
  - Support for accesses to all PCI address spaces
  - Parity support
  - Selectable hardware-enforced coherency
  - Address translation units for address mapping between host and peripheral
  - Dual address cycle supported when the device is the target
  - Internal configuration registers accessible from PCI
- Local bus controller (LBC)
  - Multiplexed 32-bit address and data
  - Eight chip selects support eight external slaves
  - Up to eight-beat burst transfers
  - 32-, 16-, and 8-bit port sizes are controlled by an on-chip memory controller
  - Three protocol engines available on a per chip select basis:
    - General-purpose chip select machine (GPCM)
    - Three user programmable machines (UPMs)
    - Dedicated single data rate SDRAM controller

  - Parity support
  - Default boot ROM chip select with configurable bus width (8-, 16-, or 32-bit)
- Programmable interrupt controller (PIC)
  - Functional and programming compatibility with the MPC8260 interrupt controller
  - Support for 8 external and 35 internal discrete interrupt sources
  - Support for one external (optional) and seven internal machine checkstop interrupt sources
  - Programmable highest priority request
  - Four groups of interrupts with programmable priority
  - External and internal interrupts directed to communication processor
  - Redirects interrupts to external $\overline{\text{INTA}}$ pin when in core disable mode
  - Unique vector number for each interrupt source
- Dual industry-standard I^2C interfaces
  - Two-wire interface
  - Multiple master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
  - System initialization data is optionally loaded from I^2C-1 EPROM by boot sequencer embedded hardware
- DMA controller
  - Four independent virtual channels
  - Concurrent execution across multiple channels with programmable bandwidth control
  - All channels accessible by local core and remote PCI masters
  - Misaligned transfer capability
  - Data chaining and direct mode
  - Interrupt on completed segment and chain
  - DMA external handshake signals: $\overline{\text{DMA\_DREQ}}$[0:3]/$\overline{\text{DMA\_DACK}}$[0:3]/$\overline{\text{DMA\_DONE}}$[0:3]. There is one set for each DMA channel. The pins are multiplexed to the parallel IO pins with other QE functions.
- DUART
  - Two 4-wire interfaces (RxD, TxD, RTS, CTS)
  - Programming model compatible with the original 16450 UART and the PC16550D
- System timers
  - Periodic interrupt timer
  - Real-time clock
  - Software watchdog timer
  - Eight general-purpose timers
- IEEE Std 1149.1™ compliant, JTAG boundary scan
- Integrated PCI bus and SDRAM clock generation

A.2 MPC8360E Features Not Present in the MPC8358E

- UCC interfaces 6 and 7
- MCC
- Two UPCs
- TDM interfaces E, F, G, and H
- DDRC2

Table A-1 shows a comparison between the MPC8358E and the MPC8360E.

Table A-1. MPC8358E and MPC8360E Comparison

	MPC8358E	MPC8360E
Core	e300	e300
CPU Speed	Up to 400MHz	Up to 667MHz
QUICC Engine Speed	Up to 400MHz	Up to 500MHz
L1 Instruction/Data Cache	32K Instruction 32K Data	32K Instruction 32K Data
Memory Controller	1x32-bit or 1x64-bit DDR1	1x64-bit or 2x32-bit DDR1
Local bus	Yes	Yes
PCI	1- 32-bit up to 66MHz	1- 32-bit up to 66MHz
Ethernet	Up to 6- 10/100 Up to 2- 10/100/1000	Up to 8- 10/100 Up to 2- 10/100/1000
HDLC/TDMs	Up to 128 / 4 (QMC ucode)	Up to 256 / 8
UTOPIA	1x31 or 1x128 MPHY	2x128 MPHY
USB	Low/Full Speed	Low/Full Speed
Security Engine	E version only	E version only
UART	Dual	Dual
I^2C	Dual	Dual
SPI	Dual	Dual
MCC	No	One
Interrupt Controller	Yes	Yes
Package	740 TBGA (37.5x37.5) 1mm pitch	740 TBGA (37.5x37.5) 1mm pitch

A.3 MPC8358E Block Diagram

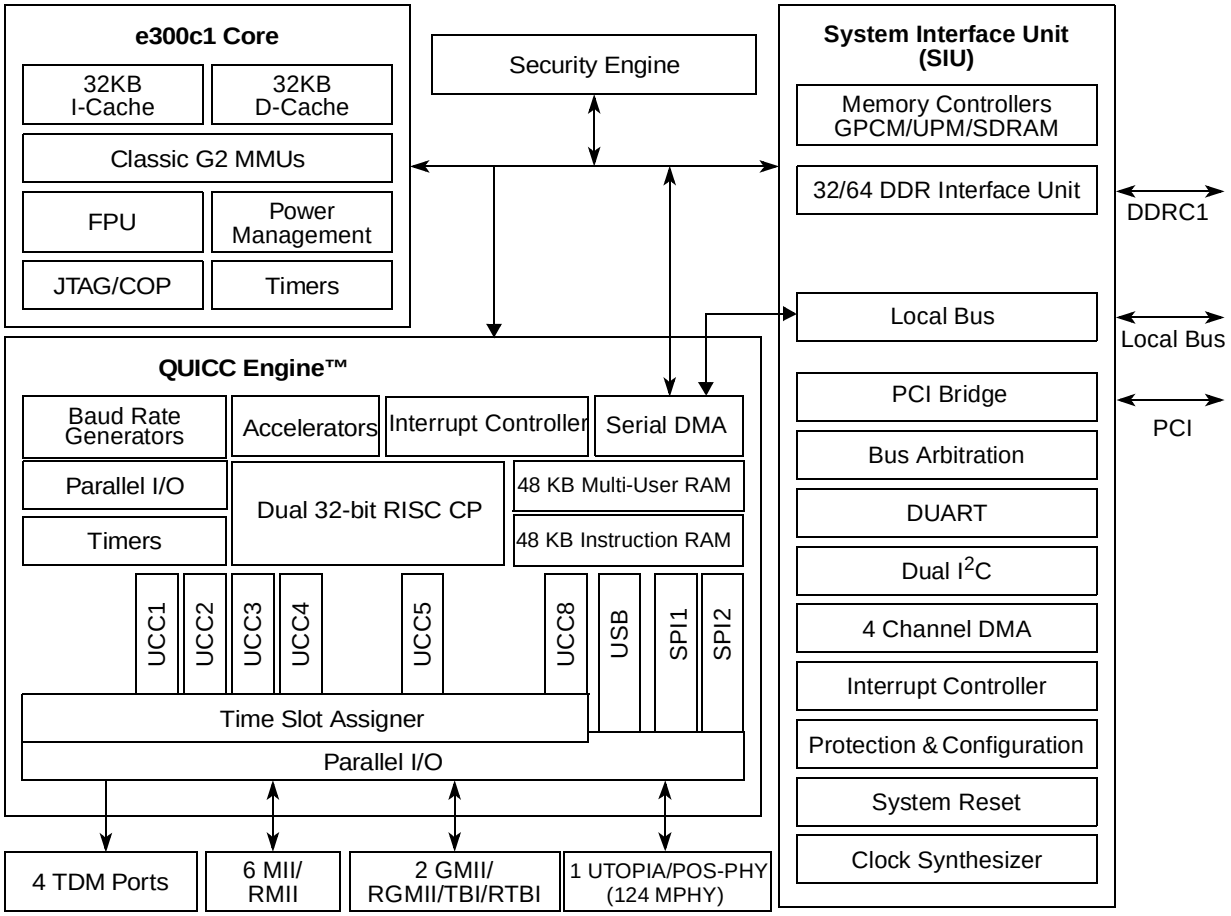


Figure A-1. MPC8358 Block Diagram

A.4 Implementation and Impact of Six UCCs

References to UCC6 and UCC7 are not applicable because the MPC8358 contains six UCCs (UCC1-5 and UCC8) instead of eight UCCs.

A.4.1 Unimplemented Signals

The following signals that support UCC6 and UCC7 are not implemented on MPC8358E:

- CMXUCR4[GR6]
- CMXUCR4[UC6]
- CMXUCR4[RU6CS]
- CMXUCR4[TU6CS]
- CMXUCR2[GR7]
- CMXUCR2[UC7]
- CMXUCR2[RU7CS]
- CMXUCR2[TU7CS]

NOTE

The alternative functionality of the associated port pins remains available, as described in Chapter 3, "Signal Descriptions," and Section 3.4, "Parallel I/O Ports."

A.4.2 Reserved Memory Map Areas

The following memory mapped areas are reserved because UCC6 and UCC7 are not available:

- 0x0_2600 - 0x0_27FF
- 0x0_3400 - 0x0_35FF

A.4.3 UCC General Setup

Although most UCC registers contain configurations bits for eight UCCs, ensure that these are only configuring UCC1-5 and UCC8. Refer to Chapter 23, "Unified Communications Controllers (UCCs)." Registers affected include CIPYCC, CMXGCR, CECR, MTC_TX_ATM_PRAM, MTC_RX_ATM_PRAM. If interrupts are used, ensure that only valid UCCEx, UCCMx, bits in CIMR and bits in CIPNR that correspond to UCC1-5 and UCC8 are manipulated and observed.

A.5 MCC

All references to MCC registers are not applicable since MPC8358 does not contain MCC. Thus, all SIRAM entries must be set for UCC (SIRAM[0] must always equal zero).

A.6 UPC Availability

Due to internal constraints, the MPC8358E supports only one UPC, either UPC1 or UPC2. Some features may become unavailable when one UPC is chosen over the other due to shared functionality on the port pins. Therefore, the user should choose either UPC1 or UPC2 depending on other features being utilized on the device. For example, some MII options are unavailable when UPC1 is chosen and PCI is unavailable when using UPC2, refer to Section 3.4, "Parallel I/O Ports" for pin muxing limitations.

The CMXUPCR register contains configurations bits for both UPCs, ensure that these are only configuring one UPC. Refer to Section 21.5.9, "CMX UPC Clock Route Register (CMXUPCR)."

NOTE

The alternative functionality of the associated port pins remains available, as described in Chapter 3, "Signal Descriptions," and Section 3.4, "Parallel I/O Ports."

A.7 Unimplemented TDM Registers

The following registers that support TDME-TDMH are not implemented on MPC8358E:

- CMXSI1CRH
- SIAMR
- SIBMR
- SICMR
- SIDMR
- SITERC
- SITFRC
- SITGRC
- SITHRC
- SIRERC
- SIRFRC
- SIRGRC
- SIRHRC
- SIGLMRL

NOTE

The alternative functionality of the associated port pins remains available, as described in Chapter 3, "Signal Descriptions," and Section 3.4, "Parallel I/O Ports."

A.8 DDR

The MPC8358 has only one DDR SDRAM memory controller (DDRC1). Thus, it can only support one 32-bit or one 64-bit bus. Therefore all references to the secondary DDR controller do not apply to the MPC8358E. In the memory map, these items do not apply:

0x00_D000 – 0x00_DFFF Secondary DDR Memory Controller

0x0_1804 SDMCSAR—Secondary DDR memory controller start address register

0x0_1844 SDMCEA

0x0_1884 SDMCAR

Appendix B
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8360E PowerQUICC II Pro Integrated Communications Processor Family Reference Manual*. Note that this list only covers the major changes.

B.1 Changes From Revision 1 to Revision 2

Section, Page No.	Changes
Global	The EFM chapter will be removed in future reference manuals. This functionality will be supported in the future via a RAM-based microcode package.
2.3/2-1	Added the following after the third paragraph: “Unless stated otherwise in a particular block, all accesses to and from the memory mapped registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.”
2-3, 2-2	Added global timer blocks to Table 2-1, “IMMR Memory Map,” and Table 2-2, “Memory Map,” as shown below. Note that the cross-references refer to this document only and are subject to change in the next revision of the document.

Table 2-1. IMMR Memory Map

Address	Use	Actual Size	Window	Cross Reference
0x00_0500–0x00_05FF	Global timers module 1	64 bytes	256 bytes	Table 2-2
0x00_0600–0x00_06FF	Global timers module 2	64 bytes	256 bytes	Table 2-2

Table 2-2. Memory Map

Offset	Register	Access	Reset	Section/Page
Global Timers Module 1				
0x0_0500	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0501–0x0_0503	Reserved, should be cleared	—	—	—
0x0_0504	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0505–0x0_050F	Reserved, should be cleared	—	—	—
0x0_0510	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0512	GTMDR2—Timer 2 global timers mode register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0514	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0516	GTRFR2—Timer 2 global timers reference register			
0x0_0518	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-15
0x0_051A	GTCPR2—Timer 2 global timers capture register			
0x0_051C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_051E	GTCNR2—Timer 2 global timers counter register			
0x0_0520	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0522	GTMDR4—Timer 4 global timers mode register			
0x0_0524	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0526	GTRFR4—Timer 4 global timers reference register			
0x0_0528	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-15
0x0_052A	GTCPR4—Timer 4 global timers capture register			
0x0_052C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_052E	GTCNR4—Timer 4 global timers counter register			
0x0_0530	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-16
0x0_0532	GTEVR2—Timer 2 global timers event register			
0x0_0534	GTEVR3—Timer 3 global timers event register			
0x0_0536	GTEVR4—Timer 4 global timers event register			
0x0_0538	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-17
0x0_053A	GTPSR2—Timer 2 global timers prescale register			
0x0_053C	GTPSR3—Timer 3 global timers prescale register			
0x0_053E	GTPSR4—Timer 4 global timers prescale register			
Global Timers Module 2				
0x0_0600	GTCFR1—Timer 1 and 2 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0601–0x0_0603	Reserved, should be cleared	—	—	—
0x0_0604	GTCFR2—Timer 3 and 4 global timers configuration register	R/W	0x00	5.8.5.1/5-10
0x0_0605–0x0_060F	Reserved, should be cleared	—	—	—
0x0_0610	GTMDR1—Timer 1 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0612	GTMDR2—Timer 2 global timers mode register			
0x0_0614	GTRFR1—Timer 1 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0616	GTRFR2—Timer 2 global timers reference register			
0x0_0618	GTCPR1—Timer 1 global timers capture register	R/W	0x0000	5.8.5.4/5-15
0x0_061A	GTCPR2—Timer 2 global timers capture register			

Table 2-2. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_061C	GTCNR1—Timer 1 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_061E	GTCNR2—Timer 2 global timers counter register			
0x0_0620	GTMDR3—Timer 3 global timers mode register	R/W	0x0000	5.8.5.2/5-14
0x0_0622	GTMDR4—Timer 4 global timers mode register			
0x0_0624	GTRFR3—Timer 3 global timers reference register	R/W	0x0000	5.8.5.3/5-15
0x0_0626	GTRFR4—Timer 4 global timers reference register			
0x0_0628	GTCPR3—Timer 3 global timers capture register	R	0x0000	5.8.5.4/5-15
0x0_062A	GTCPR4—Timer 4 global timers capture register			
0x0_062C	GTCNR3—Timer 3 global timers counter register	R/W	0x0000	5.8.5.5/5-16
0x0_062E	GTCNR4—Timer 4 global timers counter register			
0x0_0630	GTEVR1—Timer 1 global timers event register	Special	0x0000	5.8.5.6/5-16
0x0_0632	GTEVR2—Timer 2 global timers event register			
0x0_0634	GTEVR3—Timer 3 global timers event register			
0x0_0636	GTEVR4—Timer 4 global timers event register			
0x0_0638	GTPSR1—Timer 1 global timers prescale register	R/W	0x0003	5.8.5.7/5-17
0x0_063A	GTPSR2—Timer 2 global timers prescale register			
0x0_063C	GTPSR3—Timer 3 global timers prescale register			
0x0_063E	GTPSR4—Timer 4 global timers prescale register			

In same section, same table (page 2-17), removed register CAPTURE_EXT_ADDRESS from secondary DDR controller (offset 0x0_DE54).

3.1, 3-2 — In Figure 3-1, made IRQ0/MCP_IN input only.

3.4.7, 3-28 — Replaced this section, formerly entitled “RGMII pins,” with the following:

3.4.7 Gigabit Ethernet Pins

3.4.7.1 RGMII pins

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

Table 3-10. RGMII Pins

SIGNAL	UCC1 RGMII	UCC2 RGMII option 1	UCC2 RGMII option 2
CLK_IN	PC8	PC3	PC7
TxD[0]	PA3	PA17	PF5
TxD[1]	PA4	PA18	PF6

Table 3-10. RGMII Pins (continued)

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

3.4.7.2 GMII and TBI Pins

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

Table 3-11. GMII and TBI Pins

SIGNAL	UCC1 GMII/TBI	UCC2 GMII/TBI
CLK_IN	PC[8,9,10,11,14,15]	PC[2,3,6,7,15,16,17]
TxD[0]	PA3	PA17
TxD[1]	PA4	PA18
TxD[2]	PA5	PA19
TxD[3]	PA6	PA20
TxD[4]	PB6 or PC25	PB2
TxD[5]	PB7 or PC24	PB3
TxD[6	PB9 or PC23	PB5
TxD[7]	PB10 or PC22	PB8
Tx_ERR	PA8	PA22
Tx_EN	PA7	PA21
GTX_CLK	PC9	PC2
RxD[0]	PA9	PA23
RxD[1]	PA10	PA24
RxD[2]	PA11	PA25

Table 3-11. GMII and TBI Pins (continued)

SIGNAL	UCC1 GMII/TBI	UCC2 GMII/TBI
RxD[3]	PA12	PA26
RxD[4]	PA13	PA27
RxD[5]	PB1	PB12
RxD[6]	PB0	PB13
RxD[7]	PB4	PB11
Rx_ERR	PA16	PA30
Rx_DV	PA15	PA29
Rx_CLK	PA0	PA31
Rx_CLK1 (for TBI 2 clocks mode)	PC29	PC28

4.3.2.1/4-12 In Table 4-8, “Reset Configuration Word Low Bit Settings,” changed DDRCM bit description from
“The 2:1 mode is useful mostly when for a 32-bit data bus memory device.”
to
“The 2:1 mode is useful mostly for a 32-bit data bus width.”

4.3.2.2/4-16 In Figure 4-4, “Reset Configuration Word High Register (RCWHR),” and Table 4-12, “Reset Configuration Word High Bit Settings,” changed bit 29 (LALE) to be “Reserved”.

4.3.3.2.1, 4-24 Replaced the note with the following:

NOTE

When reset configuration words are loaded from an I^2C EEPROM, an I^2C serial EEPROM of extended addressing type must be used.

4.3.3.3.2, 4-28 In Table 4-24, changed the Name from ‘Reserved, should be cleared’ to Reserved’ for bits 1, 12-15, 16-19, and 20-27.

4.3.3.3.2/4-28 Reformatted Table 4-24, “Hard-Coded Reset Configuration Word High Field Values.”

4.4.3/4-31 Removed overbar from the two instances of CFG_CLKIN_DIV.

4.4.3, 4-32 Removed the first row (I^2C1) from Table 4-26, “Configurable Clock Units.”

4.5.3/4-40 In Table 4-37, “Clock Control DDR,” added a footnote for offset 0x10 as follows:

Offset 0x10 is a offset from the clock control DDR offset 0x001000.

5.4.3.4/5-27 In Table 5-39, “SPCR Bit Settings,” added note to PCIPR (bits 6-7):
“DMA has the same priority as PCI.”

5.4.3.8/5-33 In Table 5-43, “DDRCDR Field Descriptions,” revised field description bit settings for DDRCDR[DSO_NZ] and DDRCDR[DSO_PZ] as follows:

Table 5-43. DDRCDR Field Descriptions

Bits	Name	Description
2–5	DSO_PZ	DDR driver software p-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal
6–9	DSO_NZ	DDR driver software n-impedance override 0000 Half strength—Highest Z 1000 Much higher Z than nominal 1100 Higher Z than nominal 1110 Nominal impedance setting 1111 Lower Z than nominal

5.5.3, 5-37 Replace text of section (up to 5.5.4) with the following:

The WDT unit can operate in the following modes:

- WDT enable/disable mode:

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

  — WDT enable mode (SWCRR[SWEN] = 1)

    This is the default value after soft reset.

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

- WDT reset/interrupt output mode

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

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

  — Reset mode (SWCRR[SWRI] = 1).

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

  — Interrupt mode (SWCRR[SWRI] = 0).

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

- WDT prescaled/non-prescaled clock mode

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

  — Prescale mode (SWCRR[SWPR] = 1)

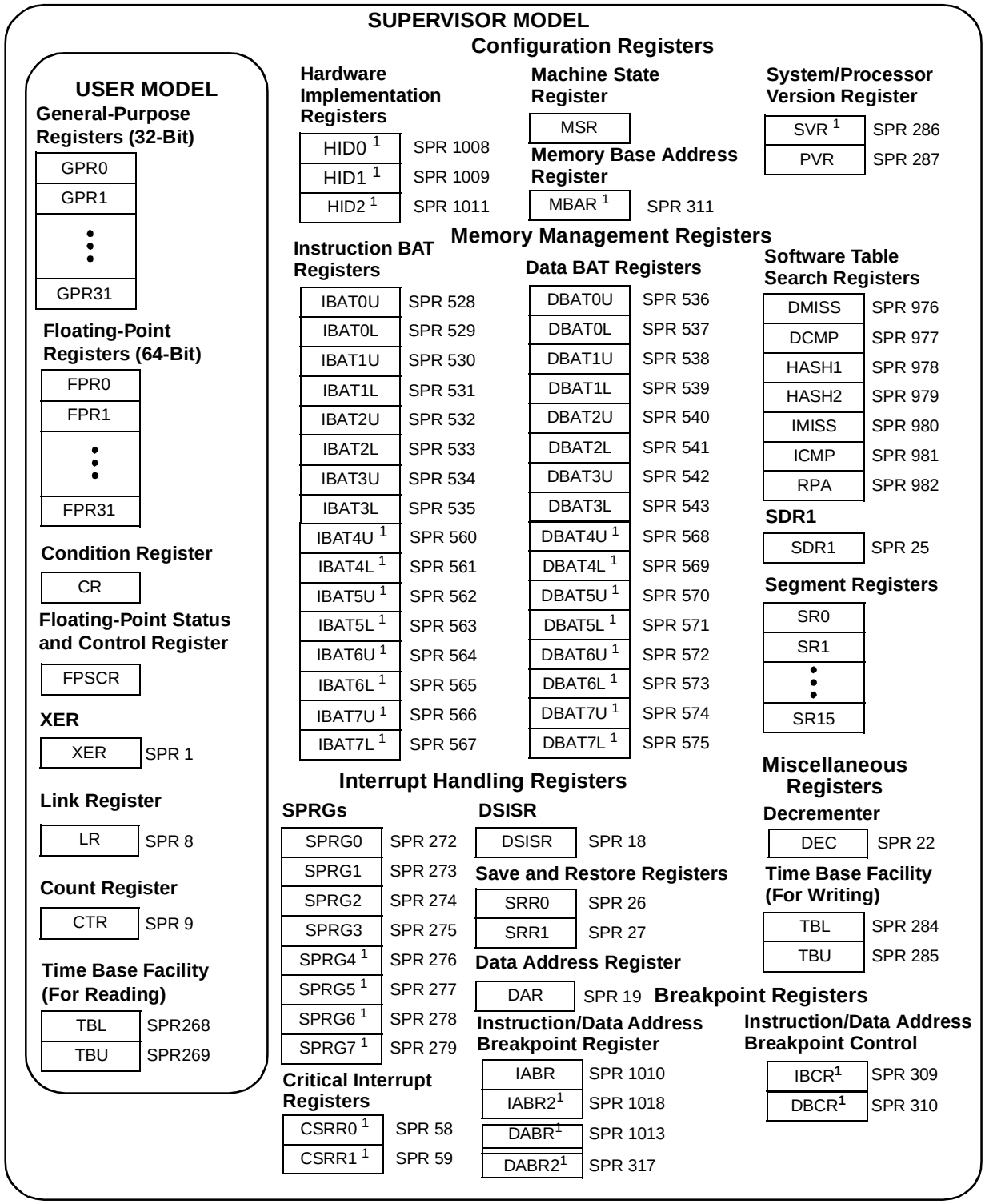
    The WDT clock is prescaled.

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

    The WDT clock is not prescaled.

5.8, 5-57	Added new Section 5.8, “Global Timers.” Any cross-references refer to this document only and are subject to change in the next revision of the reference manual. **Note that this erratum appeared in a previous revision of this addendum; however, this revision contains the following change:** **Any references to register GTGCR have been changed to GTCFR.**

7.3.1, 7-14 Added Figure 7-2 as follows:



1 These registers are e300 core implementation-specific (not defined by the PowerPC architecture).

Figure 5-1. e300 Programming Model—Registers

7.3.1.3.3, 7-20 — Added new table to show how HID0[ECLK] and HID0[SBCLK] are used to set the frequency of the *clk_out* signal.

Table 7-3. Using HID0[ECLK] and HID0[SBCLK] to Configure *clk_out*

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

7.4, 7-38 — Removed Table 7-8, “Difference Between e300 Cores”

8.5.16/8-25 — In Table 8-26, “SIFCR_H Field Descriptions” and Table 8-27, “SIFCR_L Field Descriptions”, changed INT*n* bit description of “corresponds to an external interrupt source” to “corresponds to an internal interrupt source”.

9.1/9-1, 9.2.1/9-3 — Modified text to clarify that statements about setting 8_BE when in 32-bit bus mode applies to DDR1 only.

9.3.2.1, 9-6 — Updated description of MDQS[0:8] in Table 9-3 to read as follows:

Data strobes. Inputs with read data, outputs with write data.

9.4/9-9 — In Table 9-5, “DDR Memory Controller Memory Map,” changed reset value for DDR_IP_REV2 to 0x00*nn*_00*nn* to defer implementation-specific values to register description in Section 9.4.1.17, “DDR IP Block Revision 2 (DDR_IP_REV2).”

9.4.1.6, 9-19 — Modified descriptions of TIMING_CFG_2 fields CPO and FOUR_ACT as follows:

Table 9-11. TIMING_CFG_2 Register Field Descriptions

Bits	**Name**	**Description**
4–8	CPO[1]	$\overline{\text{MCAS}}$-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

Table 9-11. TIMING_CFG_2 Register Field Descriptions (continued)

Bits	Name	Description
. . .		
26–31	FOUR_ACT	Window for four activates (t_{FAW}). This is applied to DDR2 with eight logical banks only. Must be set to 0001 for DDR1. 000000 Reserved 000001 1 cycle 000010 2 cycles 000011 3 cycles 000100 4 cycles . . . 010011 19 cycles 010100 20 cycles 010101–111111 Reserved

[1] For CPO decodings other than 00000 and 11111, ‘READ_LAT’ is rounded up to the next integer value

9.4.1.7/9-20 In Table 9-12, “DDR_SDRAM_CFG Field Descriptions,” added notes to DDR_SDRAM_CFG[RD_EN] and DDR_SDRAM_CFG[RD_EN] field descriptions stating that these fields must not both be set at the same time.

9.4.1.8, 9-23 Changed setting 01 of DDR_SDRAM_CONFIG_2[DQS_CFG] to reserved.

9.4.1.14/9-29 Replaced this section with the following:

The DDR SDRAM clock control configuration register, shown in Figure 9-15, provides a 1/4-cycle clock adjustment.

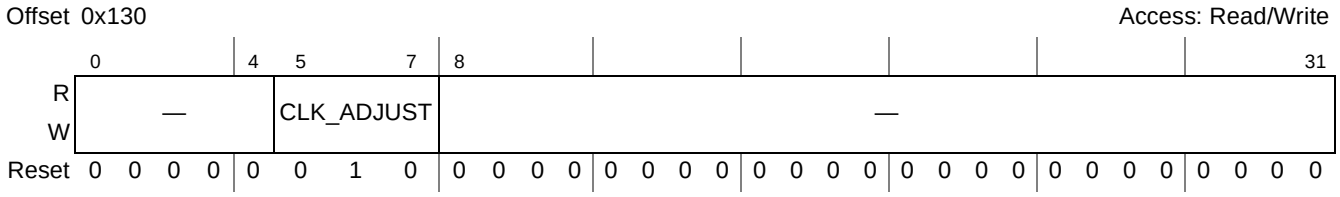


Figure 9-15. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

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 will be launched aligned with address/command 001 Clock will be launched 1/4 applied cycle after address/command 010 Clock will be launched 1/2 applied cycle after address/command 011 Clock will be launched 3/4 applied cycle after address/command 100 Clock will be launched 1 applied cycle after address/command 101–111 Reserved
8	—	Reserved, should be set to 0
9–31	—	Reserved

9.5.3/9-49	Added sentence to 6th bullet (mode register set): “For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller.”
9.6.1, 9-69	Changed DDR2 setting for DQS_CFG (in Table 9-53) to ‘Should be set to 00.’
10.3.1.3, 10-18	Updated MAR (memory address register) to show that field A is 32 bits wide (bits 0–31).
12.4.6/12-9	Changed IMISR access to mixed—two of the bits are w1c. Updated bits to reflect w1c access. Also updated memory map.
12.4.6/12-9, 12.4.7/12-11	Removed the following text from introductory paragraph of IMISR and IMIMR: “IMISR should be accessed only from the CSB and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.”
12.4.8.1/12-12	In Table 12-11, “DMAMRn Field Descriptions,” added the following to bit fields DAHE and SAHE: “The DMA does not support address hold when the external trigger mode is selected (EMSEN = 1).”
12.4.8.1, 12-13	Added the following note to DMAMR*n*[DAHE] and DMAMR*n*[SAHE]: Note that the address hold feature is not supported when external hardware control is used.
13.1, 13-2	Added the following paragraph: 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.
13.3.3.13, 13-33	Modified PIMMR[BA] as follows:

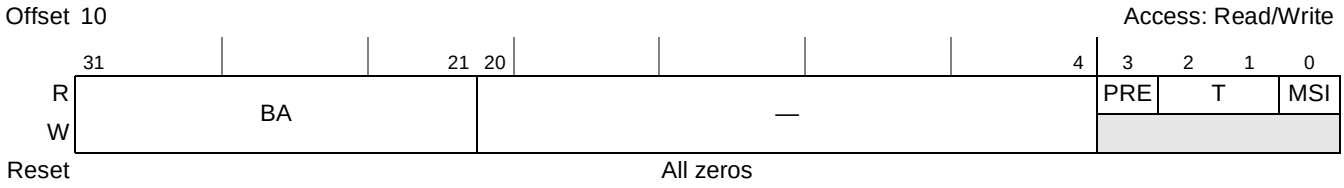


Figure 13-32. PIMMR Base Address Configuration Register

Table 13-34. 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

Table 13-34. PIMMR Base Address Configuration Register Field Descriptions (continued)

Bits	Name	Description
3	PRE	Prefetchable. Hard-wired to 0.
2–1	T	Type. Hard-wired to 00.
0	MSI	Memory space indicator. Hard-wired to 0

14.1, 14-2 Removed second sub-bullet of bullet “Master/slave logic, with DMA capability” (item read “Up to 200-MHz operation)

17.2.1, 17-2 Removed reset values (replaced with ‘—’) from all input signals (TCK, TDI, TMS) in Table 17-1, “JTAG Test Signals Summary.” Modified function column of TCK and $\overline{\text{TRST}}$. Table should appear as follows:

Table 20-1. 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

In same section, in Table 17-2, “JTAG Test—Detailed Signal Descriptions,” removed sentence “An unterminated input appears as a high signal level to the test logic due to an internal pull up resister” from description of $\overline{\text{TRST}}$.

19.2.6/19-22 In Table 19-12, interrupt number 17 should be PTP1 at vector 0b01_0001, interrupt number 19 should be PTP2 at vector 0b01_0011, and interrupt number 21 should be RTC at vector 0b01_0101.

19.3.2/19-35 In Figure 19-27, bit 0 should be PTP1, bit 2 should be PTP2, and bit 4 should be RTC.

19.3.2/19-35 Beneath Figure 19-27, add:

IEEE 1588 Interrupt (PTPx, RTC) are interrupts to the CPU that are asserted by the QUICC Engine module while it is processing an interrupt request from PTPn_TMR_PEVENT or TMR_TEVENT. This mechanism allows for the QUICC Engine RISC to interrupt the host CPU due to certain events which occur while the QUICC Engine UCC's Time Stamp Unit processes a PTP packet or upon detection of an IEEE 1588 Timer event.

20.4.1.1/20-6 In Table 20-4, add opcode 001111 PUSHSCHED, for all peripherals.

20.4.1.1/20-7 In Table 20-5, add description of opcode PUSHSCHED, “Modify the start address for the task running on a given peripheral.”

20.4.1.1.2/20-10 Directly beneath this section, add the following as **Section 20.5.1.2.3 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 request is enabled by setting CECR[17].

	0	1–6	7–14	15
Field	0	—	SNUM	1
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	CE_base + 0x00100			

	16	17	18–25	26–31
Field	—	—	—	001111
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	CE_base + 00102			

Figuren 20-3. CECR for PushSched Command

20.4.4, 20-13 In Table 20-7, the description of CERCR[CIR] was modified in a previous revision of this errata document, revision 1.3. The description should remain unchanged and appear as follows:

Table 20-7. CERCR Field Descriptions

Bit	Name	Description
4	CIR	Common Instruction RAM 0 Each of the two RISC processors has its own 24-Kbyte instruction RAM. Performance may be better since RISCs are not competing on a common resource when fetching instructions. When CIR=0, the instruction ram is seen through IADD as follows: RISC0 0x8_0000–0x8_5FFF RISC1 0x8_8000–0x8_DFFF In case the two RISCs should run the same code, the code should be duplicated in the regions for RISC0 and RISC1. 1 Both RISC processors are sharing a common 48-Kbyte instruction RAM (must be used when instruction RAM code is more than 24 Kbyte). When CIR=1, the instruction RAM is seen as a consecutive 48-Kbyte region at addresses 0x80000–0x8BFFF.

21.5.7, 21-29 — Updated description of CMXUCR3[RU2CS] in Table 21-11 by adding bit setting 1100 as follows:

1100 UCC2 receive clock is GRX_CLK

21.10.5.1/21-51 — Replaced last paragraph with the following:

Upon reaching the reference value, the corresponding GTEVRx[REF] bit is set and an interrupt is issued if GTMDRx[ORI] = 1.

21.10.5.3/21-51 — Changed Pair-cascade mode section, second paragraph; replaced last two sentences with the following:

The captures are 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.

23.3.3/23-7 — Changed “Transmit Polling Timer” to” UCC Transmit Polling Timer” and “UFPT” to “UTPT”.

24.2.4/24-7 — Changed “UFTP” to “UTPT”.

24.4.6.3/24-14 — After step 1, add, “If a protocol switch is performed from fast protocol, follow step 2 in section 24.4.6.5.

24.4.6.5/24-14 — Add the following as step 2. “If a protocol switch is performed from fast protocol: Issue the PushSched host command for UCC Transmitter (CECDR value 0x80), Issue the PushSched host command for UCC Receiver (CECDR value 0x82).

27.4.1/27-4 — In Table 27-1, “UCC Memory Map (Fast Mode),” changed “UFPT” to “UTPT”.

27.4.3/27-10 — Changed “UFTP” to “UTPT”.

29.3.1/29-3 — In Table 29-1, “UCC HDLC Register Summary,” removed UCC data synchronization register.

29.3.3/29-17 — Modified third bullet under “The collision-detection mechanism supports only:” to the following:
“Open-drain connection via port pin configuration or via external transceivers.”

30.5.1.1/30-29 In Table 30-5, “UPSMR Ethernet Field Descriptions,” replaced AUFC field description as follows:

Table 30-5. UPSMR Ethernet Field Descriptions

Bits	Name	Description
16–17	AUFC	Automatic Flow Control Enable 00 No Automatic Flow Control frames are sent by the UEC. CPU may initiate a transmit flow control frame issuing START FLOW CONTROL command. 01 In full duplex mode, the UEC sends a pause frame when RxFIFO reaches its special emergency threshold (that is, URFSET is crossed upwards). When the RxFIFO empties out below the emergency threshold (that is, URFET is crossed downwards), the UEC automatically sends pause frame with MAC parameter duration of “zero pause quantas”. It is assumed the URFET < URFSET. In half-duplex mode, when URFSET is crossed, the MAC applies back pressure algorithm on the line by sending preambles on the line when no data is available for transmit; when URFET is crossed downwards, the UEC disables the back pressure algorithm. See Section 27.5.4, “Receive Virtual FIFO Special Emergency Threshold (URFSET). 10 Reserved (not allowed) 11 In addition to the behavior described when AUFC = 01, the UEC sends 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.

30.5.1.17/30-47 Added description of UEMPR[Extension Header]:
“Can contain any user-defined data. Note that this does not extend the MAC parameter value, which would cause more delay when a flow control frame is detected.”

30.5.2.2/30-56 In Table 30-27, Offset +0, bit 15, change, “if UPSMR[IPCHK]=1,” should be “if REMODER[IPCHK]=1.”

30.6.2.6.5/30-91 In Table 30-56, “TAD Field Descriptions,” added note to Rej field description:
“For proper operation, if TAD[Rej]=1 the user must program bits TAD[VPriority,CFI,VID]=0.”

30.7.5/30-97 In Table 30-59, “UCC Statistics,” added the following sentence to the field descriptions for RBCA, RMCA, TMCA, and TBCA counters:
“Frames longer than 1518 bytes (if untagged) or 1522 bytes (if tagged) are not counted.”

30.8/30-110 In Table 30-79, removed “or transmitter error (underrun, retransmission limit reached, or late collision)” from RESTART TRANSMIT description.

31.11.3/31-32 In Table 31-25, “1588 RTC External Signals,” added parallel port signal pins columns as shown below:

Table 31-25 describes the IEEE 1588 Real Time Clock external signals:

Table 31-25. 1588 RTC External Signals

Signal name	I/O	Description	Parallel Port Signal Pins	
			Primary Option	Secondary Option
PTP_PPS1	Output	Pulse per second output signal generated by configuring the FIPER1 register. Each time the FIPER1 value is expired one RTC clock period pulse is generated	CE_PC21	CE_PE20
PTP_PPS2	Output	Pulse per second output signal generated by configuring the FIPER2 register. Each time the FIPER2 value is expired one RTC clock period pulse is generated	CE_PC17	CE_PD19
PTP_PPS3	Output	Pulse per second output signal generated by configuring the FIPER3 register. Each time the FIPER3 value is expired one RTC clock period pulse is generated	CE_PC5	CE_PF29
PTP_ALARM1	Output	Alarm output triggers, set if the timer value reaches the alarm1 register	CE_PC0	CE_PE14
PTP_ALARM2	Output	Alarm output triggers, set if the timer value reaches the alarm2 register	CE_PC23	CE_PE6
PTP_REF_CLK	Output	Divided output clock, by configuring the TMR_PRSC register	CE_PC12	CE_PE26
PTP_CLK	Input	External oscillator Real Time Clock	CE_PC30	
PTP_EXT_TRIG1	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS1L/TMR_ETSS1H	CE_PB4	CE_PC14
PTP_EXT_TRIG2	Input	Input trigger to capture timestamps. The captured timestamp is stored in TMR_ETSS2L/TMR_ETSS2H	CE_PC11	CE_PC27

36.6.4/36-18 Add the following table as Table 36-9:

Tablen 36-9. SI Mode Register Description

Bits	Name	Description
0–3	SADx	Starting address for the RAM of TDMx. These four bits define the starting address of the SI RAM section that belongs to TDMx channel. The last entry of a certain TDM is determined by the LST bit in the SI RAM entry. The user must set LST within the entries of SI RAM blocks for every TDM used i.e. before the starting address of the next TDM. 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	SDMx	SI Diagnostic Mode for all eight 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 L1GRx line is ignored. 10 Internal loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The receiver and transmitter operate normally. The data appears on the L1TXDx pin and in this mode, the L1RQx line is asserted normally. The L1GRx line is ignored. 11 Loopback control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The transmitter output (L1TXDx) and the L1RQx pin is inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines.
6–7	RFSDx.	Receive frame sync delay for all eight TDM. Determines the number of clock delays between the receive sync and the first bit of the receive frame. Even if CRTx is set, these bits do not control the delay for the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay. Use for IDL. 10 2-bit delay 01 3-bit delay
8	Reserved	Reserved. Should be cleared.

Tablen 36-9. SI Mode Register Description (continued)

Bits	Name	Description
9	CRTx	Common receive and transmit pins for all TDM. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx pins can be used as general-purpose I/O pins. 0 Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing and the transmit section uses L1TCLKx and L1TSYNCx for framing. 1 Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. Use for IDL. RFSD and TFSD are independent of one another in this mode.
10	SLx	Sync level for all TDM's. 0 The L1RSYNCx and L1TSYNCx signals are active on logic "1". 1 The L1RSYNCx and L1TSYNCx signals are active on logic "0".
11	CEx	Clock edge for all TDM's. 0 The data is transmitted on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is transmitted on the falling edge of the clock and received on the rising edge
12	FEx	Frame sync edge for all TDM's. Determines whether L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock. 0 Falling edge. Use for IDL. 1 Rising edge.
13	GMx	Grant mode for all TDM's. 0 No grant mode is supported. 1 IDL mode. A GRANT mechanism is supported if the corresponding CMXSCR[GRx] is set. The grant is a sample of L1GRx while L1TSYNCx is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel.
14–15	TFSDx	Transmit frame sync delay for all TDM's. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay 10 2-bit delay 11 3-bit delay

32.3.8, 32-112 Updated Table 32-53 as follows:

Table 32-53. V-TCT Field Descriptions

Offset	Bits	Name	Description
. . .			
0x00	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.
		VCON (mode)	Cleared by the host to enable the automatic activation in switch mode - AVCON/AVCF mechanism. When this mode is enabled, the TCT[AVCF] bit should be set. See ATM_AVCON_BASE parameter in Table 32-19 for more description.
. . .			

32.3.8.1/36-117 Replaced the last two bulleted list items with the following:

- 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 32.4.1, “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.

33.6.5.1/36-29 In Table 33-13, “UPDCx in ATM Protocol Field Descriptions,” added reserved row for bits 22-23.

34.1.1, 34-2 Added the following item to features list:

- Up to 32 Mbps sustainable aggregated throughput (for HDLC/Transparent) for normal channels and 16-bits super channels. Up to 16 Mbps for other super channels settings.

34.2.8, 34-46 Replaced beginning of description of MERL[TEO] in Table 34-23, “MCC Emergency Request Level (MERL),” with the following:

Transmit emergency offset is an offset from the beginning of the transmitter request FIFO. The offset is actually the amount of data in the request FIFO. Emergency request will be asserted when the amount of data in the request FIFO equals the amount or more.

36.1/36-1 In Figure 36-1, “SI Block Diagram,” modified note at bottom of figure to read as follows:

See Section 21.1, “Working with Peripherals in NMSI Mode,” for the connections between UCCs and pins in NMSI mode.

36.6.1, 36-12 Relocated note from after Table 36-5 to be after Table 36-4. Changed note to read as follows:

NOTE: On Backward-Compatibility

As in PowerQUICC II Pro, there is a minimal restriction of two entries per frame. There is also a minimal restriction of two bits per UCC entry if CDP or CTSP modes are set.

36.7.1, 36-32 Replaced first sentence of second paragraph with the following:

Each line of RAM consists of two 16-bit entries that define the routing control.

37.17.3, 27-58 In Table 37-27, “Tx Free Buffer Pool Parameter Table,” changed entry in bits column for first row (FBP_BASE) to read ‘0–31.’

37.18.1, 27-60 In Table 37-28, “Receive Buffer Descriptor Fields,” changed entry in bits column for last row (Time Stamp) to read ‘0–31.’

37.19.5, 37-69 Added field GBL to MPT. Updated Figure 37-45 and Table 37-33 as follows:

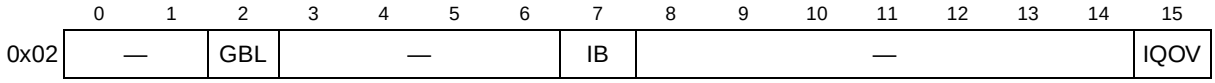


Figure 37-45. Mux Encapsulation Management Table

Table 37-33. Mux Encapsulation Management Table Field Descriptions

Offset	Bit	Size	Name	Description
0x02	2	Hword	GBL	Global. Setting GBL enables snooping of Interrupt queues. This should match BMR[GBL]. To use Global mode, SDMR[GLB_1_MSK] must also be set, see Section 19.1.8.2, “Serial DMA Mode Register (SDMR).”

37.19.2, 37-65 Replaced last sentence of CETM description in Table 37-30, “DeMux Management Table Field Description,” with the following:

See Figure 19-1 and Section 5.4.3.7, “Debug Configuration.”

37.19.7/37-74 In Figure 37-47 and Table 37-36, updated MMR by adding field Not_Act (bit 10) to indicate to the MUX process that this queue is not active. Updated Table 37-36 as follows:

Table 37-36. MMR Field Descriptions

Bits	Name	Description
8–9	—	Reserved. Should be cleared.
10	Not_Act	Not Active- This bit signals to the QE that this MUX queue is currently not active and there is no need to process a superframe for this queue. The host should clear this bit when the queue is operational.
11	—	Reserved. Should be cleared

38.3.1.1/38-15 Changed Figure 38-9, bit 14 from Reserved to RAD.

38.3.1.1/38-15 Changed Table 38-2, bit 14 from Reserved to RAD with this bit description:

Receive ATM Disable
0 Pass cells, except filtered cells, to the ATM or IMA microcode.
1 No cells are passed to the ATM or IMA microcode. The MTC will maintain the cell delineation algorithm and counters will be updated.

39.9.1, 39-31 Replaced sentence before last equation on page 39-31 with the following:

The real starting address of a RCT entry which is associated with channel code 256 is:

42.3.2.8/42-6 Bolded user initialized parameters in Table 42-1, “Global Multichannel Parameters”: MCBASE, QMCSTATE, MRBLR, Tx_S_PTR, RxPTR, GRFTHR, CRFCNT, INTBASE, INTPTR, Rx_S_PTR, TxPTR, C_MASK32, TSATRx, TSATTx, C_MASK16, and QMC_Global_Channel_specific_base.

42.3.4.1/42-16 Bolded user initialized parameters in Table 42-4, “Channel-Specific HDLC Parameters,”: TBASE, CHAMR, TSTATE, TBPTR, ZISTATE, INTMSK, RBASE, MFLR, RSTATE, RBPTR, and ZDSTATE.

42.3.4.2/42-20	Bolded user initialized parameters in Table 42-8, “Channel-Specific Transparent Parameters,”: TBASE, CHAMR, TSTATE, TBPTR, ZISTATE, INTMSK, RBASE, TMBLR, RSTATE, RBPTR, ZDSTATE, and TRNSYNC.
43.5.3.5/43-52	Change Step 8. to read as follows: Enable the corresponding link/PHY at the TC layer for example if using SAM as the TC layer set MTC_MODE[RXEN], or clear MTC_MODE[RAD] if the MTC is already in SYNC state but not passing cells up. If an external TC layer is used, set the corresponding enable bit for the external TC layer device.
43.5.3.5/43-52	Revised step 1 of the Rx steps for LDS section to clarify that after GDS failure ADD_NEW and ADD_NEW_M are correctly configured if the link that caused the failure is to be added after it reaches the working state.
43.5.3.10/43-58	Updated section to clarify that IFSD and GDS events do not require link removals.
43.5.3.10/43-58	Replaced DCB Synchronization Lost (DSL) bulleted section with the following:

- 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 43.5.3.6, “Link Removal Procedure or Section 43.5.3.7, “Link Receive Deactivation Procedure. The link can then be re-added to the group using the link addition procedure see Section 43.5.3.5, “Link Addition Procedure for a removed link or Section 43.5.3.8, “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 43.5.3.7, “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 43.5.3.8, “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 43-34. 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.

43.5.3.6.1/43-55	Change Step 8. to read as follows: Inhibit reception of cells over the dropped link by disabling the TC layer such that it does not pass any more cells to the IMA layer. Disable the TC layer, for example, if using the SAM clear MTC_MODE[RXEN]. If the TC layer is

connected on the UTOPIA bus, disable it by ensuring that it passes no more cells onto the UTOPIA bus for this link. If the application requires that the TC layer remains cell delineated (no LOCD) at this step then software should adhere to the following steps:

a) Leave MTC_MODE[RXEN] = 1

b) Set MTC_MODE[RAD]

B.2 Changes From Revision 0 to Revision 1

Section, Page No.	Changes
1.2.1, 1-8	Replace the first sentence of the second paragraph on page 1-8 with the following: The core is a superscalar processor that can issue three instructions (two plus a branch) and completes and retires as many as two instructions per clock cycle. Replace the fourth sentence of the fourth paragraph on page 1-8 with the following: The caches use a pseudo least recently used (PLRU) replacement algorithm; the TLBs use a least recently used (LRU) replacement algorithm.
2-3, 2-2	Add the following text to the beginning of this section: Reading from address locations which appear as reserved in the memory map table is not guaranteed to return predictable data. Writing to address locations which appear as reserved in the memory map table is not allowed and could lead to unpredictable behavior of the device. Reserved bits in non-reserved registers will be read as zero unless the reset value of those bits is different due to internal logic considerations. 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.
2.3, 2-5	In Table 2-2, the reset value of SWCRR (system watchdog control register) should appear as 0xFFFF_0007/0xFFFF_0003 (either value possible). A footnote was added to explain as follows: SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).
2.4, 2-35	In Table 2-4, the size of IPGIFG (interframe gap register) should be 4 bytes.
2.4, 2-36	In Table 2-4, the section “Ethernet Statistics Counters” was modified.
3.4.8, 3-28	Added POS I/O signals to Table 3-11 through Table 3-17.
3.4.8, 3-34	In Table 3-12, the input row for PB4 (CPPARBx[SELn]=11) should contain:

	UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Secondary Option)
3.4.8, 3-55	In Table 3-17, the input row for PG3 (CPPARGx[SELn]=01) should contain: UPC2:RxAddr[0] UTOPIA slave UPC2:RADR[0] POS slave (Primary Option)
4.1.2, 4-3	Add the following note to the descriptions of both CLKIN and PCI_CLK/PCI_SYNC_IN: NOTE: If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.
4.3.1.1, 4-10	In the meaning of CFG_RESET_SOURCE[0:2] (Table 4-5) for option 010, replace the last sentence with the following: Reset configuration word is loaded from an I^2C EEPROM. PCI_CLK/PCI_SYNC_IN is valid for any PCI frequency up to 66.666 MHz (range of 24–66.666 MHz).
4.3.3.1.1, 4-23	Add the following paragraph after the first paragraph: In these figures, CLOCK/32 is an internal clock. The LCLK*n* signals are not active during the power-on reset sequence.
5.2.4.1.1, 5-6	Add the following bullet below the existing one: • 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**.
5.4.1.1, 5-28	Add the following note to the description of $\overline{\text{PCI\_MODE}}$: NOTE: If PCI is not used, the device clock source must be connected to PCI_CLK/PCI_SYNC_IN, not CLKIN.
5.5.4, 5-41	In Table 5-46, the reset value of SWCRR should appear as 0xFFFF_0007/0xFFFF_0003 (either value possible). A footnote was added to explain as follows: SWCRR[SWEN] reset value directly depends on RCWHR[SWEN] (reset configuration word high).

5.5.4.1, 5-42	Added text to description of SWCRR[SWEN]: The reset value directly depends on the value of RCWHR[SWEN] bit.
8.5.19, 8-27	The offset of SCVCR should be 0x60.
8.5.20, 8-28	The offset of SMVCR should be 0x64.
9.4.1.16, 9-30	Remove this section.
9.4.1.30, 9-38	Remove this section.
14.6.1.4, 14-91	Replace paragraph three with the following: The fetch FIFO can hold up to 24 descriptor pointers at a time. When the end of the current descriptor is reached, the descriptor pointed to by the next location in the Fetch FIFO will be read to launch the next descriptor. Writing a descriptor pointer to the fetch FIFO while the FIFO is full will result in a single overflow interrupt to advise the user that the descriptor pointer was not successfully written to the fetch FIFO. The channel will continue processing and software can check the fetch FIFO counter in the crypto-channel pointer status register before attempting to re-enqueue the descriptor pointer. If a second descriptor pointer is written to the fetch FIFO before the single the single overflow error is cleared, the channel will generate a double overflow error interrupt and stop processing descriptors. The channel can be restarted by setting the Continue bit in the crypto-channel configuration register, or completely reset by writing the Reset bit in the same register.
19.1.8.9, 19-14	In Table 19-9, the description of BA should read as follows: Temporary buffer base address in multi-user RAM. The address must be 4KB aligned.
21.4, 21-7	Add the following sentence to footnote 2 of Table 21-1: The UCC1 Tx clock is also the 125MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI. Add the following sentence to footnote 4 of Table 21-1: The UCC2 Tx clock is also the 125MHz reference clock for Gigabit Ethernet when the UCC is configured for GMII/RGMII/TBI/RTBI.
27.4.2.1, 27-7	In Table 27-2, in the description of DIAG, remove the note “In Ethernet mode program MACCFG1[8] bit for loopback mode. DIAG bits have no effect..”
27.4.2.1, 27-10	In Table 27-2, in the description of ENR, change the words (in the note) “set the MACCFG1 bit 2” to “set the MACCFG1 bit 29.” In the description of ENT, change the words (in the note) “set the MACCFG1 bit 0” to “set the MACCFG1 bit 31.”
27.5.3, 27-14	The second sentence in the description of field RFET should begin “The recommended value for RFET (for most applications) is ....”
27.5.4, 27-15	The third sentence in the description of field RSFET should begin “The recommended value for RSFET ....”
29.3.3, 29-17	Replace the last two sentences of the third paragraph with the following:

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.

30.5.1.1, 30-29 — Replace the description of PTPE with the following:

PTP Enable
0 - Disable IEEE1588 assist connection to MAC I/F
1 - Enable IEEE1588 assist connection to MAC I/F

30.5.1.6, 30-38 — Add the following sentence to the description of Minimum IFG Enforcement (Table 30-12):

Zero is not a legal value.

30.5.1.8, 30-41 — Replace the description of Mgmt Clock Select in Table 30-14 with the following (note that the bit settings remain unchanged):

This field determines the clock frequency of the Mgmt Clock (CE_MDC). Its default value is 111. The source clock frequency is equal to the QUICC Engine clock divided by 16.

30.5.1.10, 30-42 — In Table 30-16, “MIIMADD Field Descriptions,” replace the text in PHY Address description with the following:

This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed, with one address reserved for internal TBI registers (corresponding to the value of TBIPA[TBIPA] whose default value is 0x00). Re-programming of TBIPA[TBIPA] to a non-zero value allows PHY address to then be set to 0.

30.5.3.3.2, 30-63 — Replace the second row of Table 30-33 (reserved area) with the following:

Offset	Bits	Name	Description (Data structure must be 32 bytes aligned).	Initialized by
0x04-0x07		Reserved	Set to zero	QE
0x08-0x09	0:31	BDRConsumerAddress	Points to the next entry in the BD ring to be executed. This field must be initialized after reset to the first entry in the BD ring.	QE
0x0A - 0x0B		Reserved		QE

30.5.3.3.3, 30-64 — In Table 30-34, add the following reference to the descriptions of NorTSRByteTime (offset 0x50) and FracSiz (offset 0x52):

Also see example calculations in Section 30.15, “Traffic Shaper Programming Considerations.”

30.5.3.5, 30-68 — In Table 30-36, add the following to the description of LossLessFCPtr:

This base address must be programmed if Lossless Flow Control is enabled in REMODER[LossLessFCEn] OR if Advanced Queue Management is enabled by programming TAD[IWCT Index]>0.

If Lossless Flow Control feature is enabled, the user must allocate 8..64 bytes for this data structure depending on the number of queues.

30.6.2.6.5, 30-90 Replace the existing Table 30-56 with the following:

Table 30-56. TAD Field Descriptions

Offset	Bits	Name	Description
0x0	0-15		See description in REMODER register. Section 30.5.3.7, “Rx Ethernet Mode Register (REMODER)
0x2	0-15	QTag	QTag (VPriority,CFI,VID) to be inserted.
0x4		Reserved	Set to zero

30.7.5, 30-97 Table 30-59 was updated extensively.

30.7.7, 30-99 Register names were modified throughout this section.

30.7.7.16, 30-107 The bits of RxDiscOv should be numbered from 16–31.

30.7.7.18, 30-109 Bits 0–15 of CMR are reserved; the non-reserved fields are in bits 16–31. (MT64 is bit 16, MT65 is bit 17, MT128 is bit 18, ...)

30.15.1.8, 30-163 In Table 30-124, the traffic rate of queue number 5 should be 1/10

31.9.1, 31-17 In Table 31-12, the names of the fields ALM2P and ALM1P should be swapped to match the register figure. In the description of setting 11 for CKSEL, change “reserved by QUICC Engine” to “NA for QUICC Engine.”

32.3.8, 32-118 Remove the fourth sentence of the second paragraph on this page (stating that the table is 64-byte aligned).

33.4.1.1, 33-19 Table 33-3 was revised to include pull up and tri-state requirements.

33.4.1.2, 33-20 Table 33-4 was revised to include tri-state requirements.

33.4.1.3, 33-21 Table 33-5 was revised to include pull up and tri-state requirements.

33.4.1.4, 33-22 Table 33-6 was revised to include tri-state requirements.

33.6.1, 33-26 Change footnote 1 of Table 33-8 to read as follows:

Note on backwards compatibility: the programing model regarding the LPB mode have changed, instead of programing the GUMR[DIAG] as in the 8260, it is done in through UPGCR[DIAG].

43.5.3.6.1, 43-55 In the TX Link Removal, updated step 8 to accomodate those cases where the TC layer must remain in SYNC state in order to send the correct state of link back to the FE.

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