MPC8272 PowerQUICC II™ Family Reference Manual

Supports
MPC8272
MPC8271
MPC8248
MPC8247

MPC8272RM
Rev. 2, 10/2005

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 2666 8080
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

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

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

Contents

Paragraph Number **Title** **Page Number**

About This Book

Chapter 1 Overview

Contents

Paragraph Number	Title	Page Number

Chapter 2
G2_LE Core

Contents

Paragraph Number | Title | Page Number

Chapter 3
Memory Map

Chapter 4
System Interface Unit (SIU)

Contents

Paragraph Number | Title | Page Number

Chapter 5
Reset

Contents

Paragraph Number | **Title** | **Page Number**

Chapter 6
External Signals

Chapter 7
60x Signals

Contents

Paragraph Number | Title | Page Number

Chapter 8
The 60x Bus

Contents

Chapter 9
PCI Bridge

Contents

Contents

Paragraph Number **Title** **Page Number**

Contents

Paragraph Number | Title | Page Number

Contents

Paragraph Number | Title | Page Number

Chapter 10
PLL and Clock Generator

Contents

Paragraph Number | Title | Page Number

Chapter 11
Memory Controller

Contents

Paragraph Number	Title	Page Number

Contents

Paragraph Number | Title | Page Number

Chapter 12
IEEE 1149.1 Test Access Port

Chapter 13
Communications Processor Module Overview

Contents

Paragraph Number | Title | Page Number

Chapter 14
Serial Interface with Time-Slot Assigner

Contents

Paragraph Number | Title | Page Number

Chapter 15
CPM Multiplexing

Chapter 16
Baud-Rate Generators (BRGs)

Chapter 17
Timers

Contents

Paragraph Number | Title | Page Number

Chapter 18
SDMA Channels and IDMA Emulation

Contents

Paragraph Number | Title | Page Number

Chapter 19
Serial Communications Controllers (SCCs)

Chapter 20
SCC UART Mode

Contents

Chapter 21
SCC HDLC Mode

Contents

Paragraph Number | Title | Page Number

Chapter 22
SCC BISYNC Mode

Chapter 23
SCC Transparent Mode

Contents

Paragraph Number | Title | Page Number

Chapter 24
SCC Ethernet Mode

Contents

Paragraph Number | Title | Page Number

Chapter 25
SCC AppleTalk Mode

Chapter 26
QMC (QUICC Multi-Channel Controller)

Contents

Paragraph Number | Title | Page Number

Chapter 27
Universal Serial Bus Controller

Contents

Paragraph Number | Title | Page Number

Chapter 28
Serial Management Controllers (SMCs)

Contents

Paragraph Number | Title | Page Number

Contents

Paragraph Number **Title** **Page Number**

Chapter 29
Fast Communications Controllers (FCCs)

Chapter 30
ATM Controller and
AAL0, AAL1, and AAL5 Protocols

Contents

Paragraph Number — Title — Page Number

Contents

Contents

Paragraph Number | Title | Page Number

Contents

Paragraph Number | Title | Page Number

Chapter 31
AAL2 Protocol

Contents

Paragraph Number | Title | Page Number

Chapter 32
Fast Ethernet Controller

Contents

Paragraph Number | Title | Page Number

Chapter 33
FCC HDLC Controller

Chapter 34
FCC Transparent Controller

Chapter 35
Serial Peripheral Interface (SPI)

Contents

Paragraph Number | Title | Page Number

Chapter 36
I^{2}C Controller

Chapter 37
Parallel I/O Ports

Contents

Paragraph Number **Title** **Page Number**

Chapter 38
Security Engine (SEC)

Contents

Paragraph Number | Title | Page Number

Contents

Paragraph Number | Title | Page Number

Contents

Contents

Paragraph Number | Title | Page Number

Appendix A
Register Quick Reference Guide

Appendix B
Revision History

Glossary of Terms and Abbreviations

Figures

Figures

Figures

Figure Number | Title | Page Number

Figures

Figures

Figure Number | Title | Page Number

Figures

Figure Number | Title | Page Number

Figures

Figure Number | Title | Page Number

Figures

Figure Number | Title | Page Number

Figures

Figure Number **Title** **Page Number**

Figures

Figures

Figure Number	Title	Page Number

Figures

Figures

Figure Number | Title | Page Number

Figures

Figure Number	Title	Page Number

Figures

Figures

Figures

Figures

Figure Number | Title | Page Number

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Table Number	Title	Page Number

Tables

Tables

Tables

Tables

Tables

Tables

Table Number | Title | Page Number

About This Book

The MPC8272 is a versatile communications processor that integrates on one chip a high-performance PowerPC™ RISC microprocessor, a very flexible system integration unit, encryption hardware, and many communications peripheral controllers that can be used in a variety of applications, particularly in communications and networking systems.

The primary objective of this manual is to help communications system designers build systems using any member of the MPC8272 PowerQUICC II™ family of communications processors and to help software designers provide operating systems and user-level applications to take fullest advantage of the MPC8272.

NOTE: Devices Supported by This Manual

> This manual supports the MPC8272, the MPC8271, the MPC8248, and the MPC8247. They are collectively referred to throughout this manual as either the MPC8272 or the PowerQUICC II. Device numbers are cited only if information does not pertain to all devices.

Although this book describes aspects regarding the PowerPC™ architecture that are critical for understanding the MPC8272 core, it does not contain a complete description of the architecture. Where additional information might help the reader, references are made to *Programming Environments Manual for 32-Bit Implementation of the PowerPC Architecture, REV 2*. Refer to “Architecture Documentation” for ordering information.

The information in this book is subject to change without notice, as described in the disclaimers on the title page of this book. As with any technical documentation, it is the readers’ responsibility to be sure they are using the most recent version of the documentation. For more information, contact your sales representative.

Before Using This Manual—Important Note

Before using this manual, determine whether it is the latest revision and if there are errata or addenda. To locate any published errata or updates for this document, refer to the world-wide web at www.freescale.com.

Audience

This manual is intended for software and hardware developers and application programmers who want to develop products for the MPC8272. It is assumed that the reader has a basic understanding of computer networking, OSI layers, and RISC architecture. In addition, it is assumed that the reader has a basic understanding of the communications protocols described here. Where it is considered useful, additional sources are provided that provide in-depth discussions of such topics.

Organization

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

- Part I, “Overview,” provides a high-level description of the MPC8272, describing general operation and listing basic features.
  - Chapter 1, “Overview,” provides a high-level description of MPC8272 functions and features. It roughly follows the structure of this book, summarizing the relevant features and providing references for the reader who needs additional information.
  - Chapter 2, “G2_LE Core,” provides an overview of the MPC8272 core, summarizing topics described in further detail in subsequent chapters.
  - Chapter 3, “Memory Map,” presents a table showing where MPC8272 registers are mapped in memory. It includes cross references that indicate where the registers are described in detail.
- Part II, “Configuration and Reset,” describes start-up behavior of the MPC8272
  - Chapter 4, “System Interface Unit (SIU),” describes the system configuration and protection functions which provide various monitors and timers, and the 60x bus configuration.
  - Chapter 5, “Reset,” describes the behavior of the MPC8272 at reset and start-up.
- Part III, “The Hardware Interface,” describes external signals, clocking, memory control, and power management of the MPC8272.
  - Chapter 6, “External Signals,” shows a functional pinout of the MPC8272 and describes the MPC8272 signals.
  - Chapter 7, “60x Signals,” describes signals on the 60x bus.
  - Chapter 8, “The 60x Bus,” describes the operation of the bus used by processors that implement the PowerPC architecture.
  - Chapter 9, “PCI Bridge,” describes how the PCI bridge enables the MPC8272 to gluelessly bridge PCI agents to a host processor that implements the PowerPC architecture and how it is compliant with PCI Specification Revision 2.2.
  - Chapter 10, “PLL and Clock Generator,” describes the clocking architecture of the MPC8272.
  - Chapter 11, “Memory Controller,” describes the memory controller, which controlling a maximum of eight memory banks shared between a general-purpose chip-select machine (GPCM) and three user-programmable machines (UPMs).
  - Chapter 12, “IEEE 1149.1 Test Access Port,” describes the dedicated user-accessible test access port (TAP), which is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*.
- Part IV, “Communications Processor Module,” describes the configuration, clocking, and operation of the various communications protocols supported by the MPC8272.
  - Chapter 13, “Communications Processor Module Overview,” provides a brief overview of the CPM.
  - Chapter 14, “Serial Interface with Time-Slot Assigner,” describes the SIU, which controls system start-up, initialization and operation, protection, as well as the external system bus.
  - Chapter 15, “CPM Multiplexing,” describes the CPM multiplexing logic (CMX) which connects the physical layer—UTOPIA, MII, modem lines,

- — Chapter 16, “Baud-Rate Generators (BRGs),” describes the eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs, SCCs, and SMCs.
- — Chapter 17, “Timers,” describes the timer implementation, which can be configured as four identical 16-bit or two 32-bit general-purpose timers.
- — Chapter 18, “SDMA Channels and IDMA Emulation,” describes the two physical serial DMA (SDMA) channels on the MPC8272.
- — Chapter 19, “Serial Communications Controllers (SCCs),” describes the four serial communications controllers (SCC), which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- — Chapter 20, “SCC UART Mode,” describes the MPC8272 implementation of universal asynchronous receiver transmitter (UART) protocol that is used for sending low-speed data between devices.
- — Chapter 21, “SCC HDLC Mode,” describes the MPC8272 implementation of HDLC protocol.
- — Chapter 22, “SCC BISYNC Mode,” describes the MPC8272 implementation of byte-oriented BISYNC protocol developed by IBM for use in networking products.
- — Chapter 23, “SCC Transparent Mode,” describes the MPC8272 implementation of transparent mode (also called totally transparent mode), which provides a clear channel on which the SCC can send or receive serial data without bit-level manipulation.
- — Chapter 24, “SCC Ethernet Mode,” describes the MPC8272 implementation of Ethernet protocol.
- — Chapter 25, “SCC AppleTalk Mode,” describes the MPC8272 implementation of AppleTalk.
- — Chapter 26, “QMC (QUICC Multi-Channel Controller),” describes the QMC protocol, which emulates up to 64 logical channels within one SCC using the same time-division-multiplexed (TDM) physical interface.
- — Chapter 27, “Universal Serial Bus Controller,” describes the MPC8272’s USB controller, including basic operation, the parameter RAM, and registers.
- — Chapter 28, “Serial Management Controllers (SMCs),” describes two serial management controllers, full-duplex ports that can be configured independently to support one of three protocols—UART, transparent, or general-circuit interface (GCI).
- — Chapter 29, “Fast Communications Controllers (FCCs),” describes the MPC8272’s fast communications controllers (FCCs), which are SCCs optimized for synchronous high-rate protocols.
- — Chapter 30, “ATM Controller and AAL0, AAL1, and AAL5 Protocols,” describes the MPC8272 ATM controller, which provides the ATM and AAL layers of the ATM protocol. The ATM controller performs segmentation and reassembly (SAR) functions of AAL5, AAL1, and AAL0, and most of the common parts convergence sublayer (CP-CS) of these protocols.
- — Chapter 31, “AAL2 Protocol,” describes he functionality and data structures of ATM adaptation layer type 2 (AAL2) CPS, CPS switching, and SSSAR.
- — Chapter 32, “Fast Ethernet Controller,” describes the MPC8272’s implementation of the Ethernet IEEE 802.3 protocol.

- Chapter 33, “FCC HDLC Controller,” describes the FCC implementation of the HDLC protocol.
- Chapter 34, “FCC Transparent Controller,” describes the FCC implementation of the transparent protocol.
- Chapter 35, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface, which allows the MPC8272 to exchange data between other MPC8272 chips, the MC68360, the MC68302, the M68HC11, and M68HC05 microcontroller families, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.
- Chapter 36, “I^{2}C Controller,” describes the MPC8272 implementation of the inter-integrated circuit (I^{2}C®) controller, which allows data to be exchanged with other I^{2}C devices, such as microcontrollers, EEPROMs, real-time clock devices, and A/D converters.
- Chapter 37, “Parallel I/O Ports,” describes the four general-purpose I/O ports A–D. Each signal in the I/O ports can be configured as a general-purpose I/O signal or as a signal dedicated to supporting communications devices, such as SMCs, SCCs, and FCCs.

- Part V, “Integrated Security Engine,” describes the MPC8272’s encryption functionality.
  - Chapter 38, “Security Engine (SEC),” describes the SEC, which is designed to off load computational intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the MPC8272.
- Appendix A, “Register Quick Reference Guide,” provides a quick reference to the registers incorporated in the G2 core.

Suggested Reading

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

MPC8272 Documentation

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.freescale.com. This documentation includes technical specifications, reference materials, and detailed applications notes.

Architecture Documentation

Architecture documentation is organized in the following types of documents:

- Manuals—These books provide details about individual implementations of the PowerPC architecture and are intended to be used in conjunction with the *Programming Environments Manual*. These include the following:
  - *G2 Core Reference Manual* (Freescale order #: G2CORERM/D, REV 0)

- Programming environments manuals—These books provide information about resources defined by the PowerPC architecture that are common to processors that implement the PowerPC architecture. There are two versions, one that describes the functionality of the combined 32- and 64-bit architecture models and one that describes only the 32-bit model. The MPC8272 adheres to the 32-bit architecture definition.
    - — *Programming Environments for 32-Bit Implementations of the PowerPC Architecture*, REV 2 (Freescale order #: MPCFPE32B/D)
- *The Programmer's Pocket Reference Guide for the PowerPC Architecture*: MPCPRGREF/D—This provides an overview of registers, instructions, and exceptions for 32-bit implementations.
- Application notes—These short documents contain useful information about specific design issues useful to programmers and engineers working with Freescale's processors.

For a current list of documentation, refer to www.freescale.com.

Conventions

This document uses the following notational conventions:

Bold	Bold entries in figures and tables showing registers and parameter RAM should F037M18be initialized by the user.
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.
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 or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In certain contexts, such as in a signal encoding or a bit field, indicates a don't care.
n	Used to express an undefined numerical value
¬	NOT logical operator
&	AND logical operator
|	OR logical operator

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document. Note that the meanings for some acronyms (such as SDR1 and DSISR) are historical, and the words for which an acronym stands may not be intuitively obvious.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
A/D	Analog-to-digital
ALU	Arithmetic logic unit
ATM	Asynchronous transfer mode
BD	Buffer descriptor
BIST	Built-in self test
BPU	Branch processing unit
BRI	Basic rate interface.
BUID	Bus unit ID
CAM	Content-addressable memory
CEPT	Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations).
CMX	CPM multiplexing logic
CPM	Communication processor module
CR	Condition register
CRC	Cyclic redundancy check
CTR	Count register
DABR	Data address breakpoint register
DAR	Data address register
DEC	Decrementer register
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DSISR	Register used for determining the source of a DSI exception
DTLB	Data translation lookaside buffer
EA	Effective address
EEST	Enhanced Ethernet serial transceiver
EPROM	Erasable programmable read-only memory
FPR	Floating-point register
FPSCR	Floating-point status and control register

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
FPU	Floating-point unit
GCI	General circuit interface
GPCM	General-purpose chip-select machine
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
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LIFO	Last-in-first-out
LR	Link register
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
MSR	Machine state register
NaN	Not a number
NIA	Next instruction address
NMSI	Nonmultiplexed serial interface
No-op	No operation
OEA	Operating environment architecture
OSI	Open systems interconnection
PCI	Peripheral component interconnect

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
PCMCIA	Personal Computer Memory Card International Association
PIR	Processor identification register
PRI	Primary rate interface
PVR	Processor version register
RISC	Reduced instruction set computing
RTOS	Real-time operating system
RWITM	Read with intent to modify
Rx	Receive
SCC	Serial communication controller
SCP	Serial control port
SDLC	Synchronous Data Link Control
SDMA	Serial DMA
SI	Serial interface
SIMM	Signed immediate value
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SPR	Special-purpose register
SPRG*n*	Registers available for general purposes
SRAM	Static random access memory
SRR0	Machine status save/restore register 0
SRR1	Machine status save/restore register 1
TAP	Test access port
TB	Time base register
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
UART	Universal asynchronous receiver/transmitter
UIMM	Unsigned immediate value
UISA	User instruction set architecture
UPM	User-programmable machine

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
USART	Universal synchronous/asynchronous receiver/transmitter
USB	Universal serial bus
VA	Virtual address
VEA	Virtual environment architecture
XER	Register used primarily for indicating conditions such as carries and overflows for integer operations

PowerPC Architecture Terminology Conventions

Table ii lists certain terms used in this manual that differ from the architecture terminology conventions.

Table ii. Terminology Conventions

The Architecture Specification	This Manual
Data storage interrupt (DSI)	DSI exception
Extended mnemonics	Simplified mnemonics
Instruction storage interrupt (ISI)	ISI exception
Interrupt	Exception
Privileged mode (or privileged state)	Supervisor-level privilege
Problem mode (or problem state)	User-level privilege
Real address	Physical address
Relocation	Translation
Storage (locations)	Memory
Storage (the act of)	Access

Table iii describes instruction field notation conventions used in this manual.

Table iii. Instruction Field Conventions

The Architecture Specification	Equivalent to:
BA, BB, BT	**crb**A, **crb**B, **crb**D (respectively)
BF, BFA	**crf**D, **crf**S (respectively)
D	d
DS	ds
FLM	FM
FXM	CRM
RA, RB, RT, RS	**r**A, **r**B, **r**D, **r**S (respectively)

Table iii. Instruction Field Conventions (continued)

The Architecture Specification	Equivalent to:
SI	SIMM
U	IMM
UI	UIMM
/, //, ///	0...0 (shaded)

Part I
Overview

Intended Audience

This part is intended for readers who need a high-level understanding of the MPC8272.

Contents

This part provides a high-level description of the MPC8272, describing general operation and listing basic features.

- Chapter 1, "Overview," provides a high-level description of MPC8272 functions and features. It roughly follows the structure of this book, summarizing the relevant features and providing references for readerS who need additional information.
- Chapter 2, "G2_LE Core," provides an overview of the MPC8272 core.
- Chapter 3, "Memory Map," presents a table showing where MPC8272 registers are mapped in memory. It includes cross references that indicate where the registers are described in detail.

Conventions

This part uses the following notational conventions:

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.
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 or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In certain contexts, such as in a signal encoding or a bit field, indicates a don't care.
n	Indicates an undefined numerical value

Acronyms and Abbreviations

Table I-1 contains acronyms and abbreviations that are used in this document.

Table I-1. Acronyms and Abbreviated Terms

Term	Meaning
ATM	Asynchronous Mode
BD	Buffer descriptor
BPU	Branch processing unit
COP	Common on-chip processor
CP	Communications processor
CPM	Communications processor module
CRC	Cyclic redundancy check
CTR	Count register
DABR	Data address breakpoint register
DAR	Data address register
DEC	Decrementer register
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DTLB	Data translation lookaside buffer
EA	Effective address
FCC	Fast communications controller
FPR	Floating-point register
GPCM	General-purpose chip-select machine
GPR	General-purpose register
HDLC	High-level data link control
I^2C	Inter-integrated circuit
IEEE	Institute of Electrical and Electronics Engineers
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
LRU	Least recently used (cache replacement algorithm)
LSU	Load/store unit
MII	Media-independent interface

Table I-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
MMU	Memory management unit
MSR	Machine state register
NMSI	Nonmultiplexed serial interface
OEA	Operating environment architecture
OSI	Open systems interconnection
PCI	Peripheral component interconnect
RISC	Reduced instruction set computing
RTC	Real-time clock
RTOS	Real-time operating system
Rx	Receive
SCC	Serial communications controller
SDLC	Synchronous data link control
SDMA	Serial DMA
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TB	Time base register
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
UART	Universal asynchronous receiver/transmitter
UISA	User instruction set architecture
UPM	User-programmable machine
VEA	Virtual environment architecture

Chapter 1
Overview

The MPC8272 is a versatile communications processor that integrates on one chip a high-performance PowerPC™ RISC microprocessor, a very flexible system integration unit, encryption hardware, and many communications peripheral controllers that can be used in a variety of applications, particularly in communications and networking systems.

The MPC8272's core—the G2_LE—is an embedded variant of the MPC603e™ microprocessor with 16 Kbytes of instruction cache and 16 Kbytes of data cache. The system interface unit (SIU) consists of a flexible memory controller that interfaces to almost any user-defined memory system, a 60x-to-PCI bus bridge, and many other peripherals that make this device a complete system on one chip.

The MPC8272's communications processor module (CPM) includes most of the peripherals found in the other PowerQUICC II families (the MPC8260 and the MPC8280). The integrated security engine (SEC) supports many different encryption algorithms, allowing for high performance data encryption and authentication as required in today's SOHO/ROBO routers.

This document describes the functional operation of the MPC8272, with an emphasis on peripheral functions. Chapter 2, "G2_LE Core," is an overview of the microprocessor core; detailed information about the core can be found in the *G2 Core Reference Manual* (order number: G2CORERM).

1.1 Features

The following is an overview of the MPC8272 feature set:

- Dual-issue integer core
  - A core version of the MPC603e microprocessor
  - System core microprocessor supporting frequencies of 266–400 MHz
  - Separate 16-Kbyte data and instruction caches:
    - Four-way set associative
    - Physically addressed
    - LRU replacement algorithm
  - PowerPC architecture–compliant memory management unit (MMU)
  - Common on-chip processor (COP) test interface
  - Supports bus snooping for cache coherency
  - Floating-point unit (FPU) supports floating-point arithmetic
  - Support for cache locking
- Low power consumption
- Separate power supply for internal logic (1.5 V) and for I/O (3.3 V)

- Separate PLLs for G2_LE core and for the CPM
  - G2_LE core and CPM can run at different frequencies for power/performance optimization
  - Internal core/bus clock multiplier that provides 2:1, 2.5:1, 3:1, 3.5:1, 4:1, 5:1, 6:1 ratios
  - Internal CPM/bus clock multiplier that provides 2:1, 2.5:1, 3:1, 3.5:1, 4:1, 5:1, 6:1 ratios
- 64-bit data and 32-bit address 60x bus
  - Bus supports multiple master designs
  - Supports single transfers and burst transfers
  - 64-, 32-, 16-, and 8-bit port sizes controlled by on-chip memory controller
- 60x-to-PCI bridge
  - Programmable host bridge and agent
  - 32-bit data bus, 66 MHz, 3.3 V
  - Synchronous and asynchronous 60x and PCI clock modes
  - All internal address space available to external PCI host
  - PCI-to-60x address remapping
- System interface unit (SIU)
  - Clock synthesizer
  - Reset controller
  - Real-time clock (RTC) register
  - Periodic interrupt timer
  - Hardware bus monitor and software watchdog timer
  - IEEE 1149.1 JTAG test access port
- Eight-bank memory controller
  - Glueless interface to SRAM, page mode SDRAM, DRAM, EPROM, Flash and other user-definable peripherals
  - Byte write enables and selectable parity generation
  - 32-bit address decodes with programmable bank size
  - Three user programmable machines, general-purpose chip-select machine, and page mode pipeline SDRAM machine
  - Byte selects for 64-bit bus width (60x)
  - Dedicated interface logic for SDRAM
- Disable CPU mode
- Communications processor module (CPM)
  - Embedded 32-bit communications processor (CP) uses a RISC architecture for flexible support for communications peripherals
  - Interfaces to G2_LE core through on-chip dual-port RAM and DMA controller. (Dual-port RAM size is 16 Kbyte)
  - Serial DMA channel for receive and transmit on all serial channels
  - Parallel I/O registers with open-drain and interrupt capability

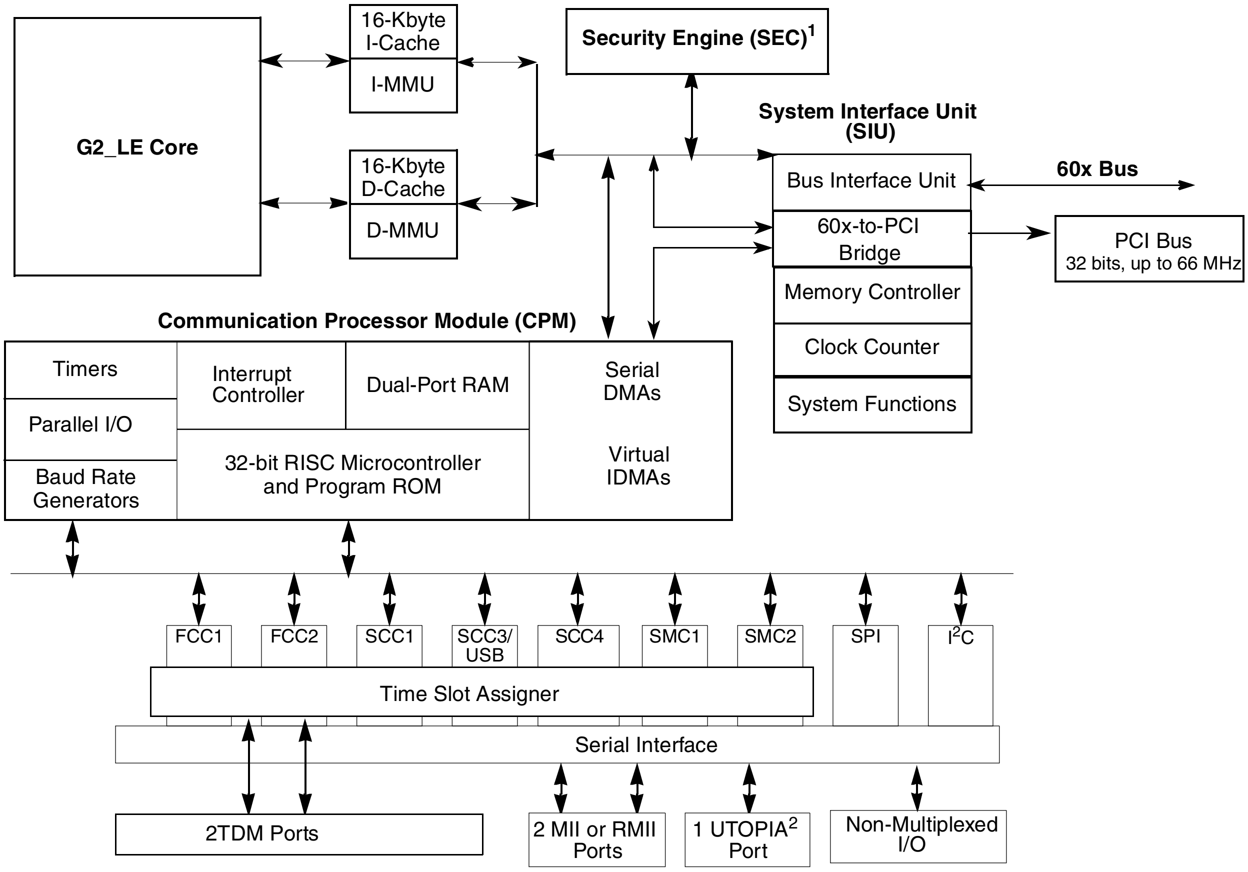
- Virtual DMA functionality executing memory to memory and memory to I/O transfers
- A fast communication controller (FCC1) supporting the following protocols:
  - 10/100-Mbit Ethernet/IEEE 802.3 CDMA/CS interface through media independent interface (MII) and reduced media independent interface (RMII)
  - ATM (MPC8272 and MPC8271 only)—Full-duplex SAR at 155 Mbps, UTOPIA interface, AAL5, AAL1, AAL0 protocols, TM 4.0 CBR, VBR, UBR, ABR traffic types, up to 64K external connections
  - Transparent
  - HDLC—Up to T3 rates (clear channel)
- A second fast communication controller (FCC2) supporting the following protocols:
  - 10/100-Mbit Ethernet/IEEE 802.3 CDMA/CS interface through media independent interface (MII) and Reduced media independent interface (RMII)
  - Transparent
  - HDLC—Up to T3 rates (clear channel)
- Three serial communications controllers (SCCs) supporting the digital portions of the following protocols:
  - Ethernet/IEEE 802.3 CDMA/CS
  - HDLC/SDLC and HDLC bus
  - Universal asynchronous receiver transmitter (UART)
  - Synchronous UART
  - Binary synchronous (BiSync) communications
  - Transparent
  - QMC (QUICC multichannel controller) up to 64 channels
    - Independent transmit and receive routing, frame synchronization.
    - Serial-multiplexed (full-duplex) input/output 2048,1544,1536-Kbps PCM highways
    - Compatible with T1/DS1 24-channel and CEPT E1 32-channels PCM highway, ISDN basic rate, ISDN primary rate and user defined
    - Sub-channeling on each time slot
    - Independent transmit and receive routing, frame synchs and clocking
    - Concatenation of any, not necessarily consecutive, time slots to channels independently for Rx/Tx
    - Supports H1,H11 and H12 channels
    - Allows dynamic allocation of channels
  - SCC3 in NMSI mode is not usable when USB is enabled.
- 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
        - Supports control, bulk, interrupt, and isochronous data transfers
        - CRC16 generation and checking
        - NRZI encoding/decoding with bit stuffing
        - Supports both 12- and 1.5-Mbps data rates (automatic generation of preamble token and data rate configuration). Note that low-speed operation requires an external hub.
        - Flexible data buffers with multiple buffers per frame
        - Supports local loopback mode for diagnostics (12 Mbps only)
    - USB slave mode
        - 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
- Two serial management controllers (SMCs)
    - Provide management for BRI devices as general-circuit interface (GCI) controllers in time-division-multiplexed (TDM) channels
    - Transparent
    - UART (low-speed operation)
- One serial peripheral interface identical to the MPC860 SPI
- One I^{2}C controller (identical to the MPC860 I^{2}C controller)
    - Microwire compatible
    - Multiple-master, single-master, and slave modes
- Up to two TDM interfaces
    - 1,024 bytes of SI RAM
    - Bit or byte resolution
    - Independent transmit and receive routing, frame synchronization.
    - Supports T1, CEPT, T1/E1, T3/E3, pulse code modulation highway, ISDN basic rate, ISDN primary rate, Freescale interchip digital link (IDL), general circuit interface (GCI), and user-defined TDM serial interfaces
- Eight independent baud rate generators and 20 input clock pins for supplying clocks to FCC, SCC, and SMC serial channels
- Four independent 16-bit timers that can be interconnected as two 32-bit timers

- PCI bridge
  - PCI Specification Revision 2.2 compliant and supports frequencies up to 66 MHz
  - On-chip arbitration
  - Support for PCI to 60x memory and 60x memory to PCI streaming
  - PCI host bridge or peripheral capabilities
  - Includes four DMA channels for the following transfers:
    - PCI-to-60x to 60x-to-PCI
    - 60x-to-PCI to PCI-to-60x
    - PCI-to-60x to PCI-to-60x
    - 60x-to-PCI to 60x-to-PCI
  - Includes all of the configuration registers (which are automatically loaded from the EPROM and used to configure the MPC8272) required by the PCI standard as well as message and doorbell registers
  - Supports the I_2O standard
  - Hot-Swap friendly (supports the Hot Swap Specification as defined by PICMG 2.1 R1.0 August 3, 1998)
  - Support for 66-MHz, 3.3-V specification
  - 60x-PCI bus core logic that uses a buffer pool to allocate buffers for each port
- Integrated encryption hardware that supports the following algorithms:
  - DES, 3DES, AES, RC-4 protocols
  - MD-5, SHA-160, SHA-256 hash calculation and HMAC
  - Public key hardware acceleration (PKHA)
  - Random number generator (RNG)

1.2 Architecture Overview

The MPC8272 has two external buses to accommodate bandwidth requirements from the high-speed system core and very fast communications channels. Figure 1-1 shows the block diagram of the superset MPC8272 device. Features that are device- or silicon-specific are noted.



Notes:
[1] MPC8272 and MPC8248 only.
[2] MPC8272 and MPC8271 only.

Figure 1-1. Block Diagram

Both the system core and the CPM have an internal PLL, which allows independent optimization of the frequencies at which they run. The system core and CPM are both connected to the 60x bus.

1.2.1 G2_LE Core

The G2_LE core is derived from the MPC603e microprocessor with power management modifications. The core is a high-performance low-power implementation of the family of reduced instruction set computer (RISC) microprocessors. The G2_LE core implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits. The G2_LE cache provides snooping to ensure data coherency with other masters. This helps ensure coherency between the CPM and the system core.

The core includes 16 Kbytes of instruction cache and 16 Kbytes of data cache. It has a 64-bit split-transaction external data bus, which is connected directly to the external MPC8272 pins.

The G2_LE core has an internal common on-chip (COP) debug processor. This processor allows access to internal scan chains for debugging purposes. It is also used as a serial connection to the core for emulator support.

The G2_LE core can be disabled. In this mode, the MPC8272 functions as a slave peripheral to an external core or to another MPC8272 device with its core enabled.

1.2.2 System Interface Unit (SIU)

The SIU consists of the following:

- A 60x-compatible parallel system bus configurable to 64-bit data width. The MPC8272 supports 64-, 32-, 16-, and 8-bit port sizes. The MPC8272 internal arbiter arbitrates between internal components that can access the bus (system core, PCI bridge, CPM, SEC and one external master). This arbiter can be disabled, and an external arbiter can be used if necessary.
- The PCI bus can be programmed as a host or as an agent. The PCI bus can be configured to run synchronously or asynchronously to the 60x bus. The MPC8272 has an internal PCI bridge with an efficient 60x-to-PCI DMA for memory block transfers.
- Memory controller supporting eight memory banks that can be allocated to the system bus. The memory controller is an enhanced version of the MPC860 memory controller. It supports three user-programmable machines. Besides all MPC860 features, the memory controller also supports SDRAM with page mode and address data pipeline.
- Supports JTAG controller IEEE 1149.1 test access port (TAP)
- A bus monitor that prevents 60x bus lock-ups, a real-time clock, a periodic interrupt timer, and other system functions useful in embedded applications

1.2.3 Communications Processor Module (CPM)

The CPM contains features that allow the MPC8272 to excel in a variety of applications targeted mainly for networking and telecommunication markets.

The CPM is a superset of the MPC860 PowerQUICC CPM, with enhancements on the CP performance and additional hardware and microcode routines that support high bit rate protocols like ATM (up to 155-Mbps full duplex) and fast Ethernet (100-Mbps full duplex).

The following list summarizes the major features of the CPM:

- The CP is an embedded 32-bit RISC controller that does not affect the performance of the G2_LE core. The CP handles the lower layer tasks and DMA control activities, leaving the G2_LE core free to handle higher layer activities. The CP has an instruction set optimized for communications, but can also be used for general-purpose applications, relieving the system core of small and often repeated tasks.
- Two serial DMA (SDMA) that can do simultaneous transfers, optimized for burst transfers to the 60x bus
- Two full-duplex, serial fast communications controllers (FCCs) supporting ATM (155 Mbps) protocol through UTOPIA2 interface, IEEE 802.3 and fast Ethernet protocols, HDLC up to E3

rates (45 Mbps) and totally transparent operation. Each FCC can be configured to transmit fully transparent and receive HDLC, or vice-versa.

- Three full-duplex serial communications controllers (SCCs) supporting IEEE802.3/Ethernet, high-level synchronous data link control, HDLC, local talk, UART, synchronous UART, BISYNC, and transparent.
- Two full-duplex serial management controllers (SMC) supporting GCI, UART, and transparent operations
- Serial peripheral interface (SPI) and I2C bus controllers
- Time-slot assigner (TSA) that supports multiplexing of data from any of the three SCCs, two FCCs, and two SMCs

1.2.4 Security Engine (SEC)

The MPC8272's integrated security engine (SEC) is designed to offload computational intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the MPC8272. It is optimized to process all algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i.

1.3 Software Compatibility

As much as possible, the MPC8272 CPM features were made similar to those of the previous PowerQUICC devices (MPC860). The code flow ports easily from previous devices to the MPC8272, except for new protocols supported by the MPC8272.

Although many registers are new, most registers retain the old status and event bits, so an understanding of the programming models of the MC68360, MPC860, or MPC85015 is helpful. Note that the MPC8272 initialization code requires changes from the MPC860 initialization code (Freescale provides reference code).

1.3.1 Signals

Figure 1-2 shows MPC8272 signals grouped by function. Note that many of these signals are multiplexed (parallel I/O ports), and this figure does not indicate how these signals are multiplexed.

NOTE

> A bar over a signal name indicates that the signal is active low—for example, $\overline{\text{BB}}$ (bus busy). 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 TSIZ[0–3] (transfer size signals), which are referred to as asserted when they are high and negated when they are low.

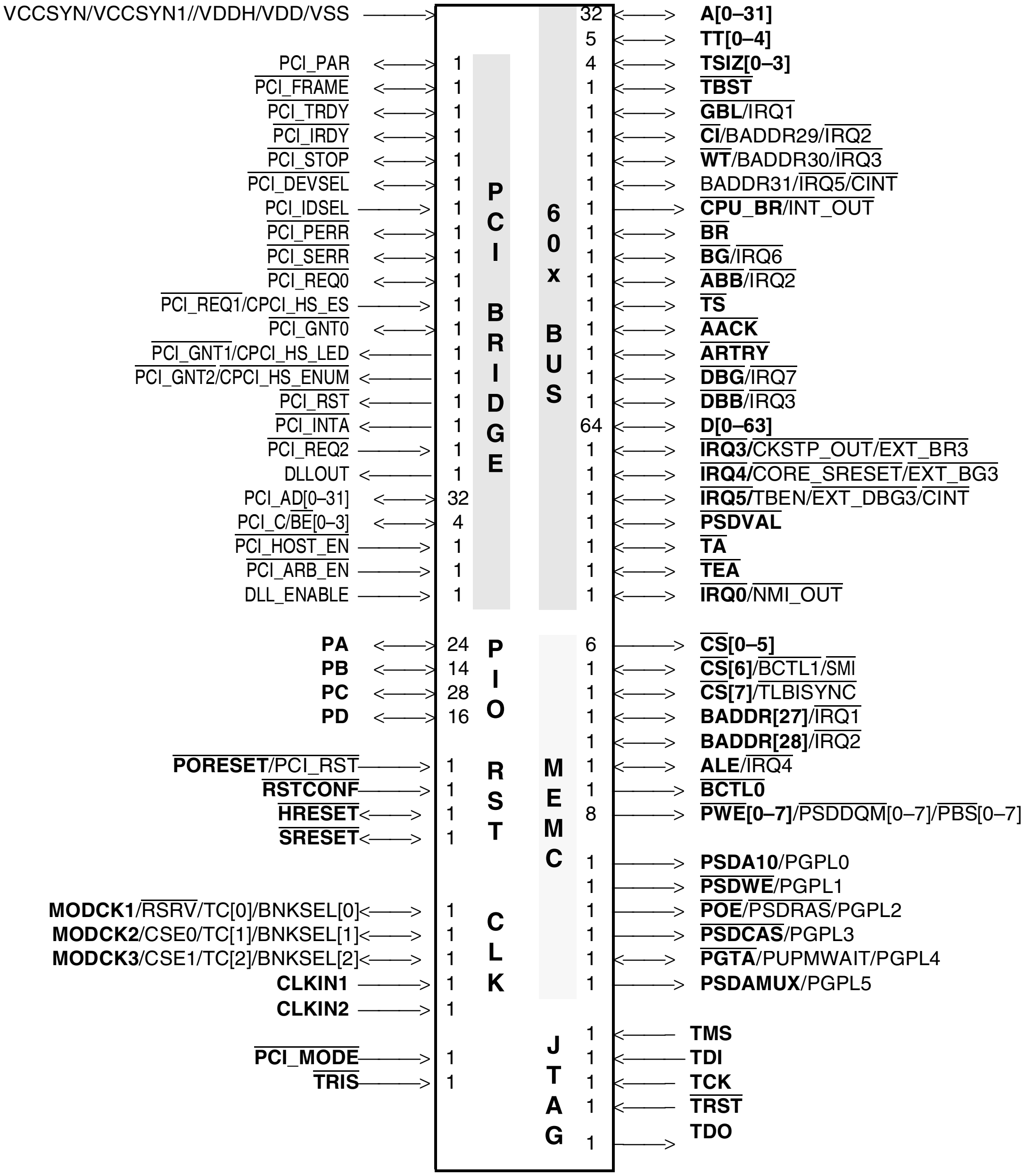


Figure 1-2. Signal Names

1.4 Differences Between the MPC860 and the MPC8272

The following MPC860 features are not included in the MPC8272:

- On-chip crystal oscillators (must use external oscillator)
- 4-MHz oscillator (input clock must be at the bus speed)
- Low power (stand-by) modes
- Battery-backup real-time clock (must use external battery-backup clock)
- BDM (COP offers most of the same functionality)

- True little-endian mode
- PCMCIA interface
- Infrared (IR) port
- Multiply and accumulate (MAC) block in the CPM
- Centronics port (PIP)
- Pulse-width modulated outputs
- SCC Ethernet controller option to sample 1 byte from the parallel port when a receive frame is complete
- Parallel CAM interface for SCC (Ethernet)

1.5 Serial Protocol Table

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

Table 1-1. MPC8272 Serial Protocols

Port	Port			
	FCC	SCC	SMC	USB
ATM (Utopia)[1]	√			
100BaseT	√			
10BaseT	√	√		
HDLC	√	√		
HDLC_BUS		√		
Transparent	√	√	√	
UART		√	√	
DPLL		√		
QMC (Multichannel)		√		
USB				√

[1] Not on the MPC8248 or the MPC8247.

1.6 MPC8272 Configurations

The MPC8272 offers flexibility in configuring the device for specific applications. The functions mentioned in the above sections are all available in the device, but not all of them can be used at the same time. This does not imply that the device is not fully activated in any given implementation: The CPM architecture has the advantage of using common hardware resources for many different protocols and applications. Two physical factors limit the functionality in any given system—pinout and performance.

1.6.1 Pin Configurations

Some pins have multiple functions. Choosing one function may preclude the use of another. Information about multiplexing constraints can be found in Chapter 15, "CPM Multiplexing," and Chapter 37, "Parallel I/O Ports."

1.6.2 Serial Performance

Serial performance depends on a number of factors:

- Serial rate versus CPM clock frequency for adequate sampling on serial channels
- Serial rate and protocol versus CPM clock frequency for CP protocol handling
- Serial rate and protocol versus bus bandwidth
- Serial rate and protocol versus system core clock for adequate protocol handling

Please consult the CPM Performance Tool for performance estimates.

The following configurations shows two examples with maximum bit rate, and variable packet size.

1.7 Application Examples

The MPC8272 can be configured to meet many system application needs, as described in the following sections and shown in Figure 1-3 through Figure 1-4.

1.7.1 Examples of Communication Systems

Communication examples:

- Small office router
- LAN-to-WAN bridge router
- General-purpose controller

1.7.1.1 Small Office Router

Figure 1-3 shows a small office router configuration.

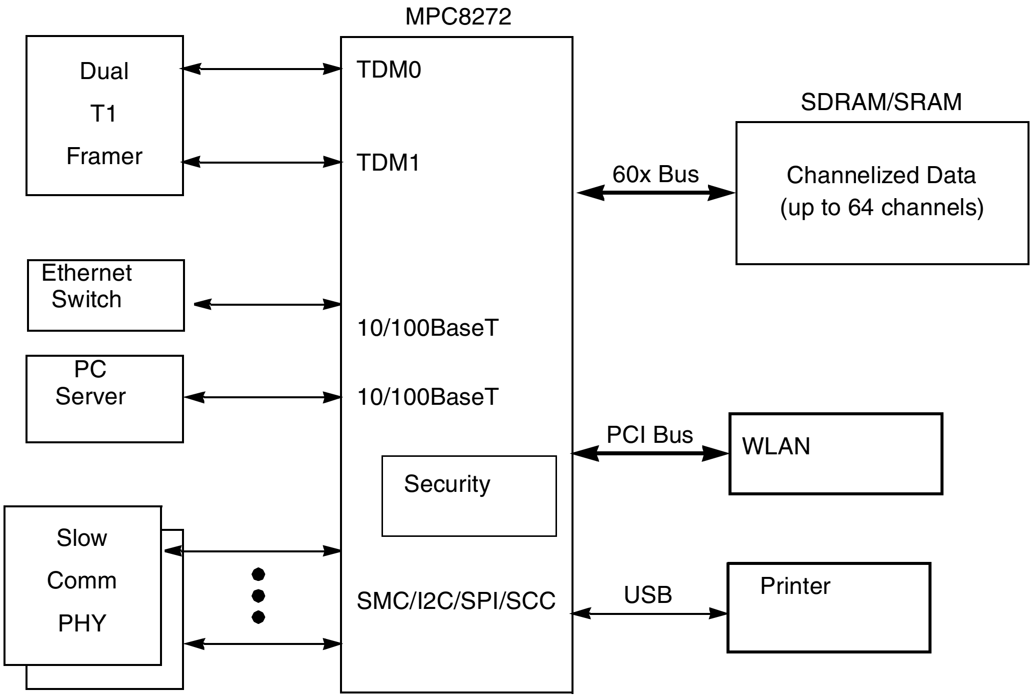


Figure 1-3. Small Office Router Configuration

In this application, the MPC8272 is connected to two TDM interfaces channelizing up to 64 channels and using two of the three SCCs. Each TDM port supports up to 32 channels. This example has two RMII ports for 10/100 BaseT LAN connections. They are used to interface to local or network devices—either a PC server or an external L2 Ethernet switch. The PCI can be used to further expand the local area network to wireless LAN. The USB port can be used to interface to a printer.

1.7.1.2 ADSL Small Office Router

Figure 1-4 shows an ADSL router configuration, which is similar to the previous example.

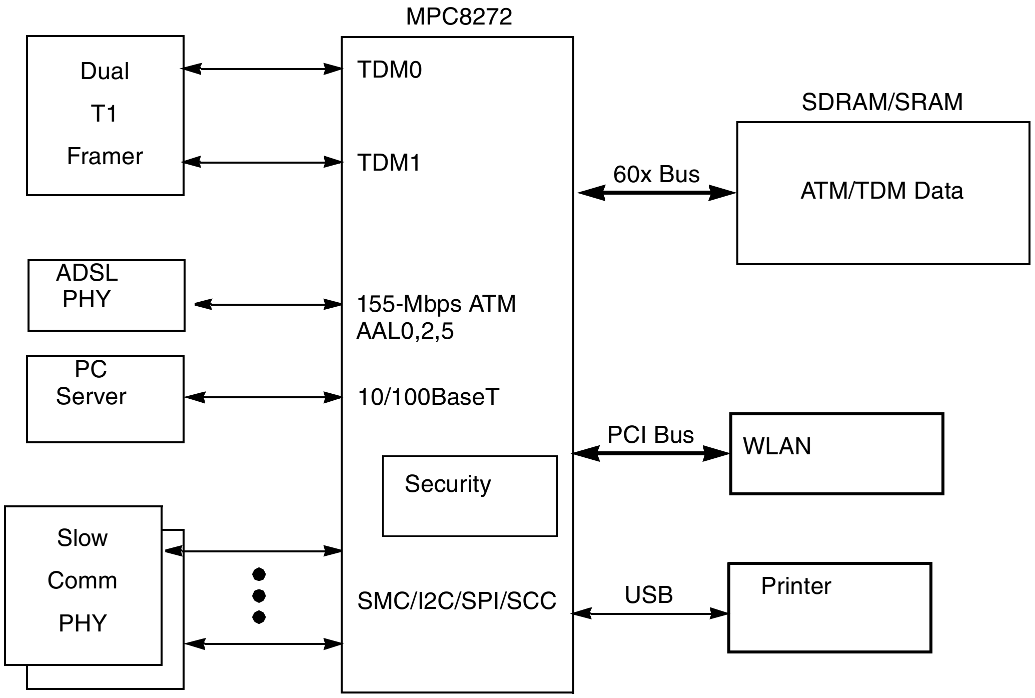


Figure 1-4. ADSL Router Configuration

1.7.1.3 General-Purpose Controller

The PowerQUICC is very suitable for general-purpose controller applications. Its flexible communication capabilities allow for many different applications.

1.7.2 Bus Configurations

The following sections describe the following possible bus configurations:

- Section 1.7.2.1, “Basic System”
- Section 1.7.2.2, “High-Performance Communication”
- Section 1.7.2.3, “High-Performance System Microprocessor”
- Section 1.7.2.4, “MPC8272 as PCI Agent”

1.7.2.1 Basic System

In the basic system configuration, shown in Figure 1-5, the MPC8272 core is enabled and uses the 64-bit 60x data bus. The CP can store large data frames in local memory without interfering with the operation of the system core.

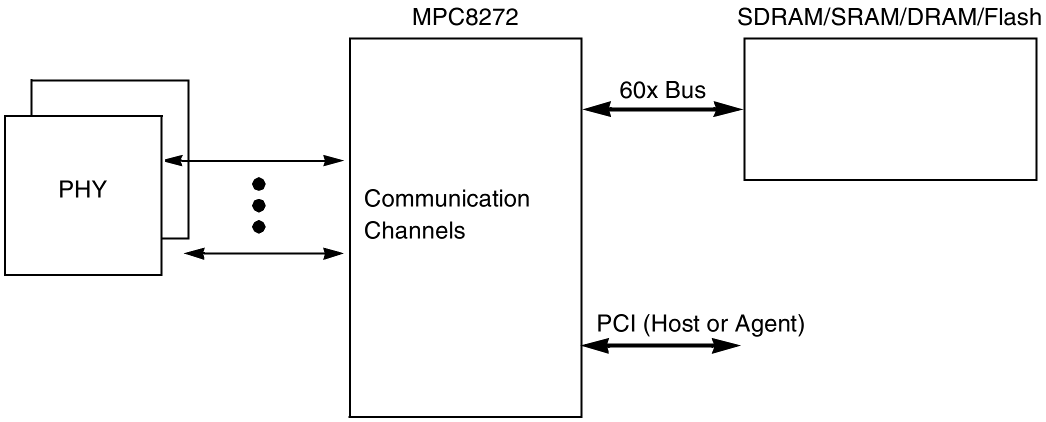


Figure 1-5. Basic System Configuration

1.7.2.2 High-Performance Communication

Figure 1-6 shows a high-performance communication configuration.

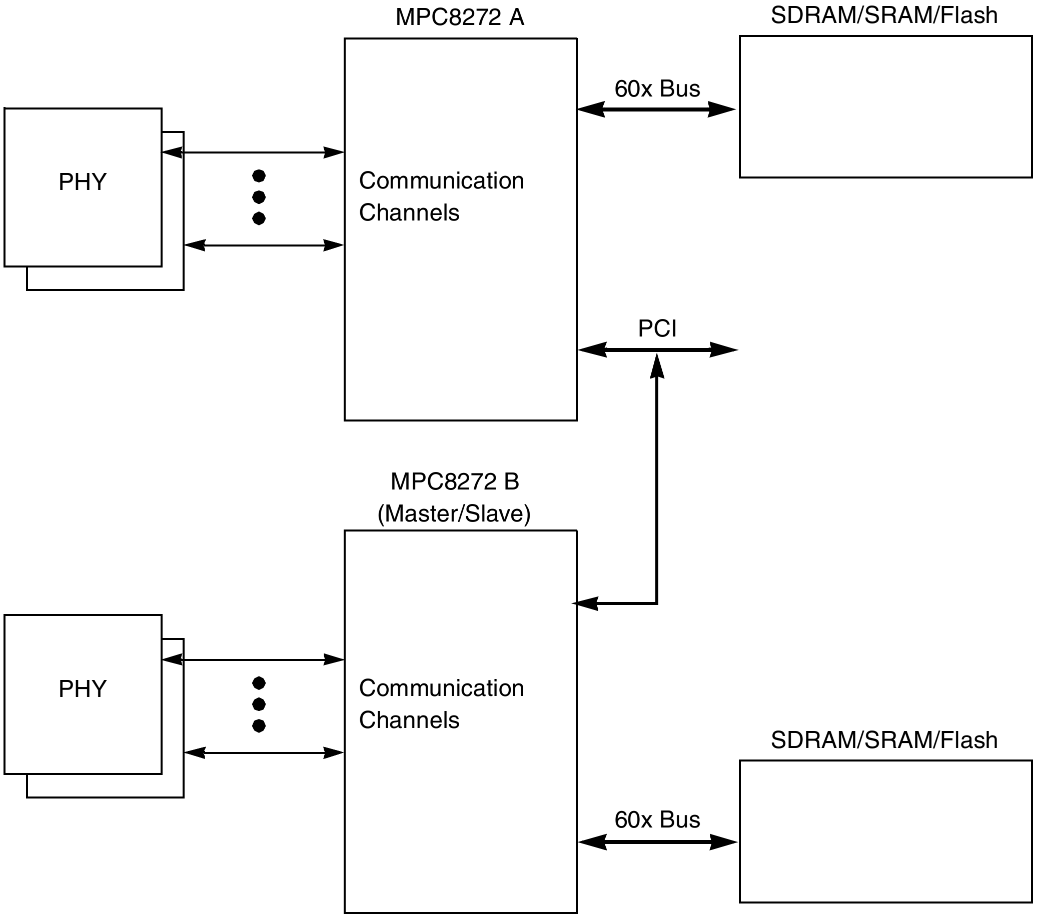


Figure 1-6. High-Performance Communication

It is possible to connect two PowerQUICC II devices either through the PCI port (as depicted in the figure above) or through the 60x bus.

1.7.2.3 High-Performance System Microprocessor

Figure 1-7 shows a configuration with a high-performance system microprocessor (MPC750).

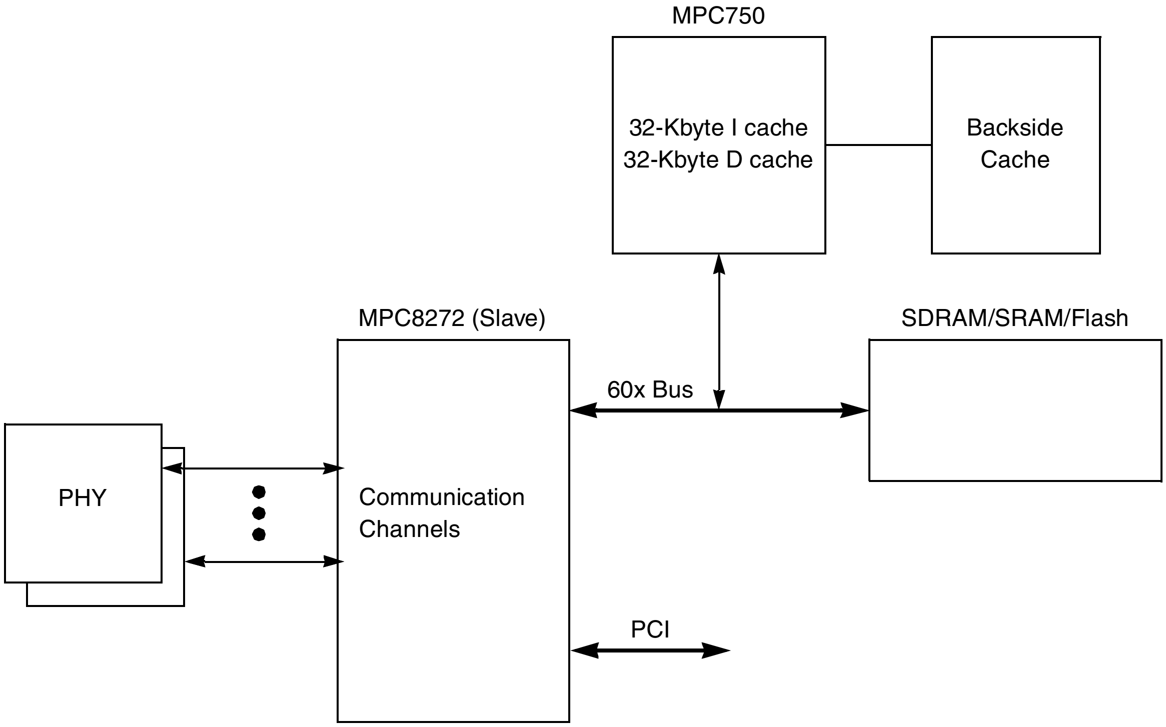


Figure 1-7. High-Performance System Microprocessor Configuration

In this system, the G2_LE core internal is disabled and an external high-performance microprocessor is connected to the 60x bus.

1.7.2.4 MPC8272 as PCI Agent

Figure 1-8 shows the configuration when the MPC8272 acts as the PCI agent.

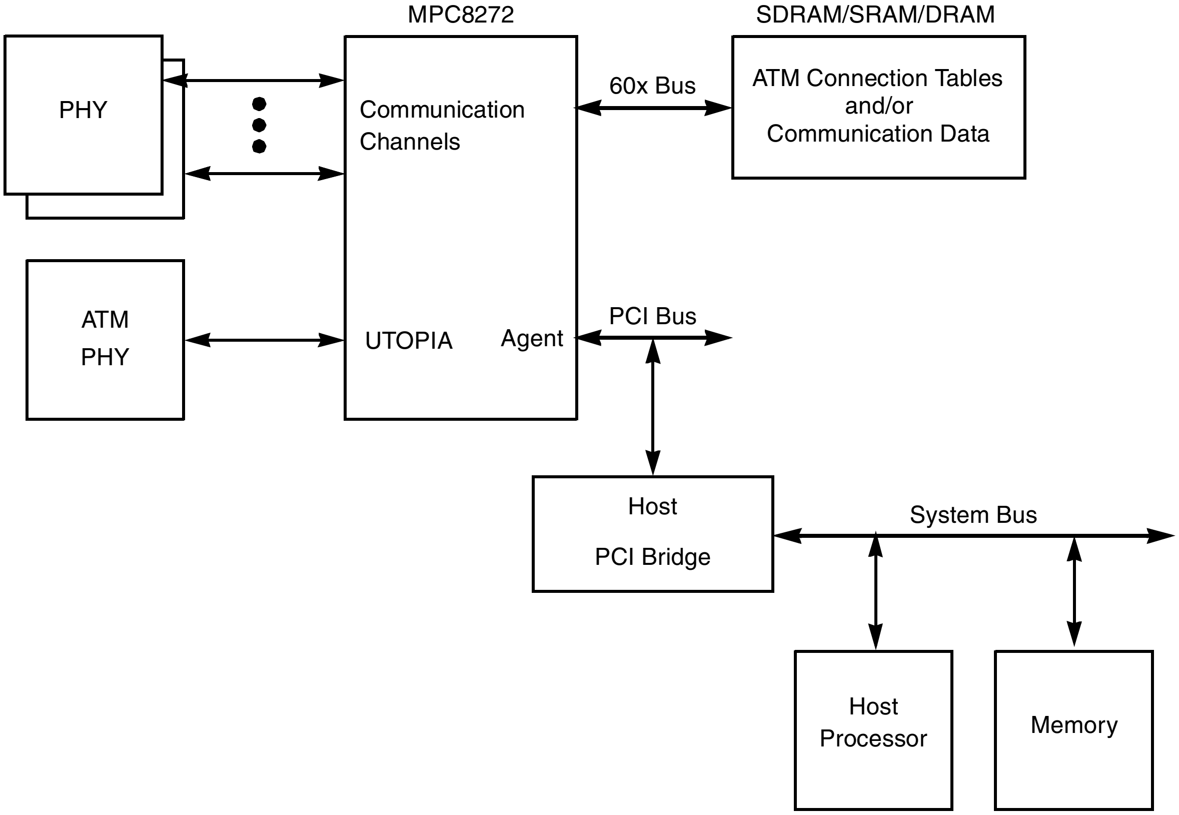


Figure 1-8. MPC8272 as PCI Agent

In this system, the PowerQUICC II is a PCI agent on an I/O card and the PCI host resides on the PCI bus. An external PCI bridge is used to connect the host to the PCI bus. The internal PCI bridge in the PowerQUICC II is used to bridge between the PCI bus and the 60x bus on the PowerQUICC II.

Chapter 2
G2_LE Core

The MPC8272 contains an embedded G2_LE processor core, which is a derivative of the G2 core and the original MPC603e PowerPC microprocessor design. This chapter provides an overview of the basic functionality of the G2_LE processor core. For detailed information regarding the processor, refer to the following:

- *G2 Core Reference Manual*
- *The Programming Environments for 32-Bit Implementations of the PowerPC Architecture*

This section describes the details of the processor core, provides a block diagram showing the major functional units, and briefly describes how those units interact.

MPC827-specific implementation includes most of the G2_LE features. The unimplemented features are described in the Section 2.2, “G2_LE Core Features.”

The signals associated with the processor core are described individually in Chapter 7, “60x Signals.” Chapter 8, “The 60x Bus,” describes how those signals interact.

2.1 Overview

The processor core is a low-power implementation of the family reduced instruction set computing (RISC) microprocessors that implement the PowerPC architecture. The processor core implements the 32-bit portion of the PowerPC architecture, which supports 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

Figure 2-1 is a block diagram of the processor core.

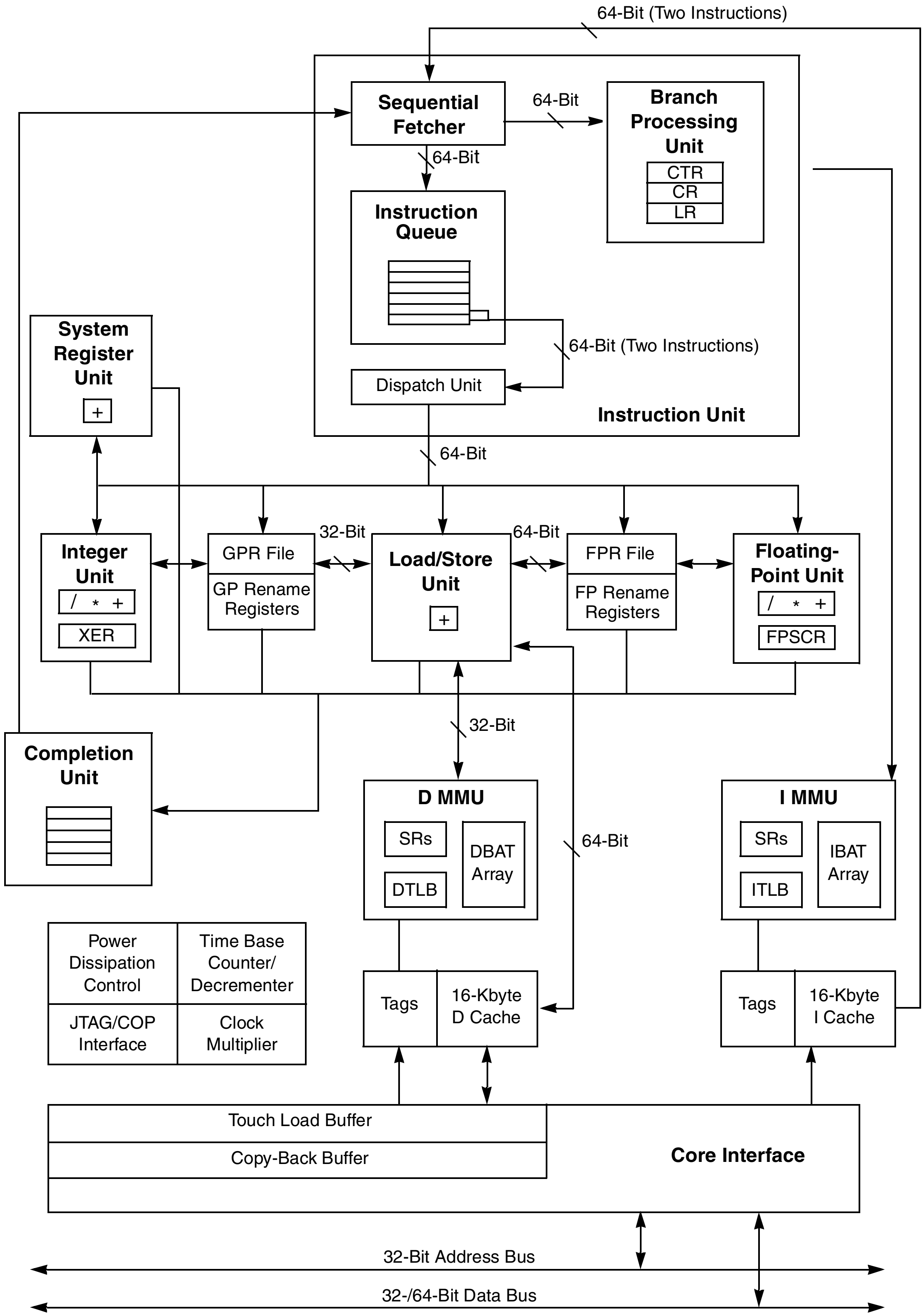


Figure 2-1. MPC8272 Integrated Processor Core Block Diagram

The processor core is a superscalar processor that can issue and retire as many as three instructions per clock. Instructions can execute out of order for increased performance; however, the processor core makes completion appear sequential.

The processor core integrates five execution units—an integer unit (IU), a floating-point unit (FPU), a branch processing unit (BPU), a load/store unit (LSU), and a system register unit (SRU). The ability to execute 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. On the processor core, the FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle. The processor core provides hardware support for all single- and double-precision floating-point operations for most value representations and all rounding modes.

The processor core supports integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits. The 32 architecturally-defined floating point registers (FPRs) can be used to hold 32, 64-bit operands that can in turn be transferred to and from the 32 general-purpose registers (GPRs), which can hold 32, 32-bit operands.

The processor core provides separate on-chip, 16-Kbyte, four-way set-associative, physically addressed caches for instructions and data and on-chip instruction and data memory management units (MMUs). The MMUs contain 64-entry, two-way set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged virtual memory address translation and variable-sized block translation. The TLBs and caches use a least recently used (LRU) replacement algorithm. The processor core 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 PowerPC architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

As an added feature to the MPC603e core, the MPC8272 can lock the contents of 1–3 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 processor core has a 60x bus that incorporates a 64-bit data bus and a 32-bit address bus. The processor core supports single-beat and burst data transfers for memory accesses and supports memory-mapped I/O operations.

2.2 G2_LE Core Features

This section describes the major features of the processor core:

- High-performance, superscalar microprocessor
  - — As many as three instructions issued and retired per clock cycle
  - — As many as four instructions in execution per clock cycle
  - — Single-cycle execution for most instructions
  - — Pipelined FPU for all single-precision and most double-precision operations

- Five independent execution units and two register files
  - BPU featuring static branch prediction
  - A 32-bit IU
  - Fully IEEE 754–compliant FPU for both single- and double-precision operations
  - LSU for data transfer between data cache and GPRs and FPRs
  - SRU that executes condition register (CR), special-purpose register (SPR), and integer add/compare instructions
  - Thirty-two GPRs for integer operands
  - Thirty-two FPRs for floating-point operands. They also can be used for general operands using floating-point load and store operations.
- High instruction and data throughput
  - Zero-cycle branch capability (branch folding)
  - Programmable static branch prediction on unresolved conditional branches
  - BPU that performs CR lookahead operations
  - Instruction fetch unit capable of fetching two instructions per clock from the instruction cache
  - A six-entry instruction queue that provides lookahead capability
  - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
  - 16-Kbyte data cache—Four-way set-associative, physically addressed; LRU replacement algorithm
  - 16-Kbyte instruction cache—Four-way set-associative, physically addressed; LRU replacement algorithm
  - Cache write-back or write-through operation programmable on a per page or per block basis
  - Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size
  - A 64-entry, two-way set-associative ITLB
  - A 64-entry, two-way set-associative 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 32- or 64-bit, split-transaction external data bus with burst transfers
  - Support for one-level address pipelining and out-of-order bus transactions
  - Critical interrupt exception support
  - Hardware support for misaligned little-endian accesses
- Integrated power management
  - Power-saving mode: doze
  - Automatic dynamic power reduction when internal functional units are idle

- Deterministic behavior and debug features
  - On-chip cache locking options for the instruction and data caches (1–3 ways or the entire cache contents can be locked)
  - In-system testability and debugging features through JTAG and boundary-scan capability

Features supported by the G2_LE core not present on the MPC8272:

- True little-endian mode for compatibility with other true little-endian devices
- Nap and sleep power-saving modes

Figure 2-1 shows how the execution units—IU, 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.

The processor core provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing is handled in the instruction unit. The MMUs translate addresses for cache or external memory accesses.

2.2.1 Instruction Unit

As shown in Figure 2-1, the instruction unit, which contains a fetch unit, instruction queue, dispatch unit, and the 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 extracts branch instructions from the fetcher and uses static branch prediction on unresolved conditional branches to allow the instruction unit to fetch instructions from a predicted target instruction stream while a conditional branch is evaluated. The BPU folds out branch instructions for unconditional branches or conditional branches unaffected by instructions in progress in the execution pipeline.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these instructions are to be executed in the BPU, they are decoded but not issued. Instructions to be executed by the IU, LSU, and SRU are issued and allowed to complete up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and instruction execution continues without interruption on the predicted path. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

2.2.2 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 2-1, 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. Reservation stations at the IU, LSU, and SRU facilitate instruction dispatch to those units. The dispatch unit checks for source and destination register dependencies, determines dispatch serializations, and inhibits subsequent instruction dispatching as required. Section 2.7, “Instruction Timing,” describes instruction dispatch in detail.

2.2.3 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, instructions are fetched 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 CR. The BPU calculates the return pointer for subroutine 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 other instructions.

2.2.4 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.

In addition to the BPU, the processor core provides three other execution units and a completion unit, which are described in the following sections.

2.2.4.1 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. Thirty-two general-purpose registers are provided to support integer operations. Stalls due to contention for GPRs are minimized by the automatic allocation of rename registers. The processor core writes the contents of the rename registers to the appropriate GPR when integer instructions are retired by the completion unit.

2.2.4.2 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 processor core to efficiently implement multiply and multiply-add operations. The FPU is pipelined so that single-precision instructions and double-precision instructions can be issued back-to-back. Thirty-two floating-point registers are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by the automatic allocation of rename registers. The core writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The processor core supports all IEEE 754 floating-point data types (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software exception routines.

2.2.4.3 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 translated in program order; however, the actual memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering where needed.

Cacheable loads, when free of data dependencies, execute in an out-of-order manner with a maximum throughput of one per cycle and 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. Store operations do not occur until a predicted branch is resolved. They remain in the store queue until the completion logic signals that the store operation will be completed to memory.

The processor core executes store instructions with a maximum throughput of one per cycle and a three-cycle total latency. The time required to perform the actual load or store operation varies depending on whether the operation involves the cache, system memory, or an I/O device.

2.2.4.4 System Register Unit (SRU)

The SRU executes various system-level instructions, including condition register logical operations and move to/from special-purpose register instructions, and also executes integer add/compare instructions. Because SRU instructions affect modes of processor operation, most SRU instructions 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 the instruction completes.

2.2.5 Completion Unit

The completion unit tracks instructions from dispatch through execution, and then retires, or completes them in program order. Completing an instruction commits the processor core to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the processor core must recover from a mispredicted branch or any exception.

Instruction state and other information required for completion is kept in a first-in-first-out (FIFO) queue of five completion buffers. A single completion buffer is allocated for each instruction once it enters the dispatch unit. An available completion buffer is a required resource for instruction dispatch; if no completion buffers are available, instruction dispatch stalls. A maximum of two instructions per cycle are completed in order from the queue.

2.2.6 Memory Subsystem Support

The processor core supports cache and memory management through separate instruction and data MMUs (IMMU and DMMU). The processor core also provides dual 16-Kbyte instruction and data caches, and 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 subsections.

2.2.6.1 Memory Management Units (MMUs)

The processor core's MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gbytes (2^{32}) of physical memory (referred to as real memory in the PowerPC architecture specification) for instructions 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. 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 the effective addresses for instruction fetching.

The MMUs translate effective addresses and enforce 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.

2.2.6.2 Cache Units

The processor core provides independent 16-Kbyte, four-way set-associative instruction and data caches. The cache block size is 32 bytes. The caches are designed to adhere to a write-back policy, but the processor core allows control of cacheability, write policy, and memory coherency at the page and block levels. The caches use a least recently used (LRU) replacement algorithm.

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.

2.3 Programming Model

The following subsections describe the PowerPC instruction set and addressing modes.

2.3.1 Register Set

This section describes the register organization in the processor core as defined by the three programming environments of the PowerPC architecture—the user instruction set architecture (UISA), the virtual environment architecture (VEA), and the operating environment architecture (OEA), as well as the G2 core implementation-specific registers. Full descriptions of the basic register set defined by the PowerPC architecture are provided in Chapter 2 in *The Programming Environments Manual*.

The PowerPC architecture defines register-to-register operations for all arithmetic instructions. Source data for these instructions is accessed from the on-chip registers or is provided as an immediate value embedded in the opcode. The three-register instruction format allows specification of a target register distinct from the two source registers, thus preserving the original data for use by other instructions and reducing the number of instructions required for certain operations. Data is transferred between memory and registers with explicit load and store instructions only.

Figure 2-2 shows the complete MPC8272 register set and the programming environment to which each register belongs. This figure includes both the PowerPC register set and the MPC8272-specific registers.

Note that some registers common to other processors that implement the PowerPC architecture may not be implemented in the MPC8272's processor core. Unsupported SPR values are treated as follows:

- Any **mtspr** with an invalid SPR executes as a no-op.
- Any **mfspr** with an invalid SPR causes boundedly undefined results in the target register.

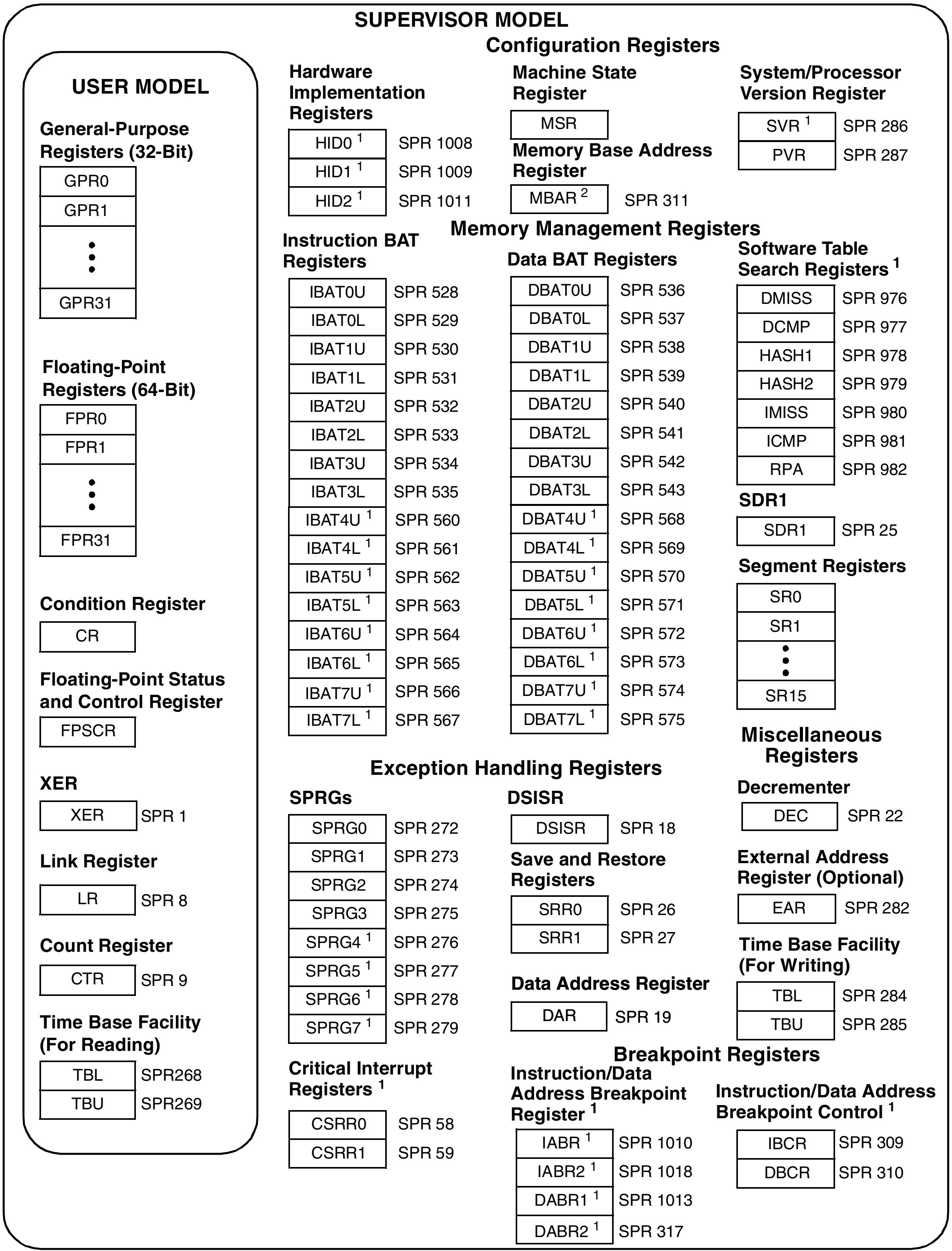
Conversely, some SPRs in the processor core may not be implemented or may not be implemented in the same way as in other processors that implement the PowerPC architecture.

2.3.1.1 PowerPC Register Set

The PowerPC UISA registers, shown in Figure 2-2, can be accessed by either user- or supervisor-level instructions. The general-purpose registers (GPRs) and floating-point registers (FPRs) are accessed through instruction operands. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as the **mtspr** and **mfspr** instructions) or implicit as part of the execution (or side effect) of an instruction. Some registers are accessed both explicitly and implicitly.

The number to the right of the register name indicates the number that is used in the syntax of the instruction operands to access the register (for example, the number used to access the XER is 1). For more information on the PowerPC register set, refer to Chapter 2 in *The Programming Environments Manual*.

Note that the reset value of the MSR exception prefix bit (MSR[IP]), described in the *G2 Core Reference Manual*, is determined by the CIP bit in the hard reset configuration word in the MPC8272. This is described in Section 5.4.1, "Hard Reset Configuration Word."



[1] These implementation-specific registers may not be supported by other PowerPC processors or processor cores.

Figure 2-2. MPC8272 Programming Model—Registers

2.3.1.2 MPC8272-Specific Registers

The set of registers specific to the MPC603e are also shown in Figure 2-2. Most of these are described in the *G2 Core Reference Manual* and are implemented in the MPC8272 as follows:

- MMU software table search registers: DMISS, DCMP, HASH1, HASH2, IMISS, ICMP, and RPA. These registers facilitate the software required to search the page tables in memory.
- IABR1 and IABR2. These registers facilitate the setting of instruction address breakpoints.
- DABR1 and DABR2. These registers facilitate the setting of data address breakpoints.
- IBCR and DBCR. These registers give further control to the instruction and data address breakpoints.

The hardware implementation-dependent registers (HIDx) are implemented differently in the MPC8272, and they are described in the following subsections.

2.3.1.2.1 Hardware Implementation-Dependent Register 0 (HID0)

Figure 2-3 shows the MPC8272 implementation of HID0.

0	1	2	3	4		6	7	8	9	10	11	12		14	15
EMCP	—	EBA	EBD	—			PAR	DOZE	—	STOP	DPM	—			NHR
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICE	DCE	ILOCK	DLOCK	ICFI	DCFI	—		IFEM	—		FBIOB	ABE	—		NOOPTI

Figure 2-3. Hardware Implementation Register 0 (HID0)

Table 2-1 shows the bit definitions for HID0.

Table 2-1. HID0 Field Descriptions

Bits	Name	Description
0	EMCP	Enable machine check input pin 0 The assertion of the $\overline{\text{MCP}}$ does not cause a machine check exception. 1 Enables the entry into a machine check exception based on assertion of the $\overline{\text{MCP}}$ input, detection of a cache parity error, detection of an address parity error, or detection of a data parity error. Note that the machine check exception is further affected by MSR[ME], which specifies whether the processor checkstops or continues processing.
1	—	Reserved
2	EBA	Enable/disable 60x bus address parity checking 0 Prevents address parity checking. 1 Allows a address parity error to cause a checkstop if MSR[ME] = 0 or a machine check exception if MSR[ME] = 1. EBA and EBD let the processor operate with memory subsystems that do not generate parity.
3	EBD	Enable 60x bus data parity checking 0 Parity checking is disabled. 1 Allows a data parity error to cause a checkstop if MSR[ME] = 0 or a machine check exception if MSR[ME] = 1. EBA and EBD let the processor operate with memory subsystems that do not generate parity.

Table 2-1. HID0 Field Descriptions (continued)

Bits	Name	Description
4–6	—	Reserved
7	PAR	Disable precharge of $\overline{\text{ARTRY}}$. 0 Precharge of $\overline{\text{ARTRY}}$ enabled 1 Alters bus protocol slightly by preventing the processor from driving $\overline{\text{ARTRY}}$ to high (negated) state, allowing multiple $\overline{\text{ARTRY}}$ signals to be tied together. If this is done, the system must restore the signals to the high state.
8	DOZE	Doze mode enable. Operates in conjunction with MSR[POW]. [1] 0 Doze mode disabled. 1 Doze mode enabled. Doze mode is invoked by setting MSR[POW] after this bit is set. In doze mode, the PLL, time base, and snooping remain active.
9	—	Reserved, should be cleared.
10	STOP	Stop mode enable. Operates in conjunction with MSR[POW]. [1] 0 Stop mode disabled. 1 Stop mode enabled. Sleep mode is invoked by setting MSR[POW] while this bit is set. When this occurs, the processor asserts $\overline{\text{QREQ}}$ to indicate that it is ready to enter sleep mode. The main MPC8272's PLL remains active and all the internal clocks—including the core's clock—stop.
11	DPM	Dynamic power management enable. [1] 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.
12–14	—	Reserved
15	NHR	Not hard reset (software-use only)—Helps software distinguish a hard reset from a soft reset. 0 A hard reset occurred if software had previously set this bit. 1 A hard reset has not occurred. If software sets this bit after a hard reset, when a reset occurs and this bit remains set, software can tell it was a soft reset.
16	ICE	Instruction cache enable [2] 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 accesses are propagated to the bus as single-beat transactions. For those transactions, however, $\overline{\text{CI}}$ reflects the original state determined by address translation regardless of cache disabled status. ICE is zero at power-up. 1 The instruction cache is enabled
17	DCE	Data cache enable [2] 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 accesses are propagated to the bus as single-beat transactions. For those transactions, however, $\overline{\text{CI}}$ reflects the original state determined by address translation regardless of cache disabled status. DCE is zero at power-up. 1 The data cache is enabled.

Table 2-1. HID0 Field Descriptions (continued)

Bits	Name	Description
18	ILOCK	Instruction cache lock 0 Normal operation 1 Instruction cache is locked. A locked cache supplies data normally on a hit, but an access is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat, however, $\overline{\text{CI}}$ still reflects the original state as determined by address translation independent of cache locked or disabled status. To prevent locking during a cache access, an **isync** must precede the setting of ILOCK.
19	DLOCK	Data cache lock 0 Normal operation 1 Data cache is locked. A locked cache supplies data normally on a hit but an access is treated as a cache-inhibited transaction on a miss. On a miss, the transaction to the bus is single-beat, however, $\overline{\text{CI}}$ still reflects the original state as determined by address translation 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** must precede the setting of DLOCK.
20	ICFI	Instruction cache flash invalidate [2] 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 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 ICFI clears all the valid bits of the blocks and the PLRU bits to point to way L0 of each set. Once the L1 flash invalidate bits are set through an **mtspr** instruction, hardware automatically resets these bits in the next cycle (provided that the corresponding cache enable bits are set in HID0).
21	DCFI	Data cache flash invalidate [2] 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. Once the L1 flash invalidate bits are set through an **mtspr** instruction, hardware automatically resets these bits in the next cycle (provided that the corresponding cache enable bits are set in HID0).
22–23	—	Reserved
24	IFEM	Enable M bit on 60x bus for instruction fetches 0 M bit not reflected on 60x bus. Instruction fetches are treated as nonglobal on the bus. 1 Instruction fetches reflect the M bit from the WIM settings on the 60x bus.
25–26	—	Reserved
27	FBIOB	Force branch indirect on bus. 0 Register indirect branch targets are fetched normally 1 Forces register indirect branch targets to be fetched externally.
28	ABE	Address broadcast enable 0 **dcbf**, **dcbi**, and **dcbst** instructions are not broadcast on the 60x bus. 1 **dcbf**, **dcbi**, and **dcbst** generate address-only broadcast operations on the 60x bus.

Table 2-1. HID0 Field Descriptions (continued)

Bits	Name	Description
29–30	—	Reserved
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 globally.

[1] See Chapter 10, “Power Management,” of the *G2 Core Reference Manual* for more information.

[2] See Chapter 4, “Instruction and Data Cache Operation,” of the *G2 Core Reference Manual* for more information.

2.3.1.2.2 Hardware Implementation-Dependent Register 1 (HID1)

The MPC8272 implementation of HID1 is shown in Figure 2-4.

0 4	5 31
PLLCFG	—

Figure 2-4. Hardware Implementation-Dependent Register 1 (HID1)

Table 2-2 shows the bit definitions for HID1.

Table 2-2. HID1 Field Descriptions

Bits	Name	Function
0–4	PLLCFG	PLL configuration setting. These bits reflect the state of the PLL_CFG[0:4] signals.
5–31	—	Reserved

2.3.1.2.3 Hardware Implementation-Dependent Register 2 (HID2)

The processor core implements an additional hardware implementation-dependent register, shown in Figure 2-5.

0 12	13	14	15	16 18	19 23	24 26	27 31
—	HBE	—	SFP	IWLCK	—	DWLCK	—

Figure 2-5. Hardware Implementation-Dependent Register 2 (HID2)

Table 2-3 describes the HID2 fields.

Table 2-3. HID2 Field Descriptions

Bits	Name	Function
0–12	—	Reserved
13	HBE	High BAT enable. Enables the four additional pairs of BAT registers (IBAT4–IBAT7 and DBAT4–DBAT7). These BATs are accessible by the **mfspr** and **mtspr** instructions regardless of the setting of HID2[HBE].
14	—	Reserved

Table 2-3. HID2 Field Descriptions (continued)

Bits	Name	Function
15	SFP	Speed for low power. Setting SFP reduces power consumption at the cost of reducing the maximum frequency, which benefits power-sensitive applications that are not frequency-critical.
16–18	IWLCK	Instruction cache way lock. Useful for locking blocks of instructions into the instruction cache for time-critical applications that require deterministic behavior. See Section 2.4.2.3, “Cache Locking.”
19–23	—	Reserved
24–26	DWLCK	Data cache way lock. Useful for locking blocks of data into the data cache for time-critical applications where deterministic behavior is required. See Section 2.4.2.3, “Cache Locking.”
27–31	—	Reserved

2.3.1.2.4 Processor Version Register (PVR)

Software can identify the MPC8272’s processor core by reading the processor version register (PVR). The processor version number is 0x80822013.

2.3.2 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 greatly simplifies instruction pipelining.

2.3.2.1 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the memory operand is considered to wrap around from the maximum effective address to effective address 0.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

In addition to the functionality of the MPC603e, the MPC8272 has additional hardware support for misaligned little-endian accesses. Except for string/multiple load and store instructions, little-endian load/store accesses not on a word boundary generate exceptions under the same circumstances as big-endian requests.

2.3.2.2 PowerPC Instruction Set

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include arithmetic and logical instructions.
  - Integer arithmetic
  - Integer compare
  - Integer logical
  - Integer rotate and shift
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic
  - Floating-point multiply/add
  - Floating-point rounding and conversion
  - Floating-point compare
  - Floating-point status and control
- Load/store instructions—These include integer and floating-point load and store instructions.
  - Integer load and store
  - Integer load and store with byte reverse
  - Integer load and store string/multiple
  - Floating-point load and store
- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other synchronizing instructions that affect the instruction flow.
  - Branch and trap
  - Condition register logical
  - Primitives used to construct atomic memory operations (**lwarx** and **stwcx.**)
  - Synchronize
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
  - Move to/from SPR
  - Move to/from MSR
  - Instruction Synchronize
- Memory control instructions—These provide control of caches, TLBs, and segment registers.
  - Supervisor-Level Cache Management
  - User-Level Cache Management
  - Segment Register Manipulation
  - TLB Management

- The G2_LE core implements the following instructions, which are defined as optional by the PowerPC architecture:
  - External Control In Word Indexed (**eciwx**)
  - External Control Out Word Indexed (**ecowx**)
  - 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 the instructions does not indicate which execution unit executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. The PowerPC architecture uses instructions that are four 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. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. It also provides for word and double-word operand loads and stores between memory and a set of 32 floating-point registers (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 written back to the target location with separate instructions. Decoupling arithmetic instructions from memory accesses increases throughput by facilitating pipelining.

Processors that implement the PowerPC architecture follow the program flow when they are 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 exception may cause one of several components of the system software to be invoked.

2.3.2.3 MPC8272 Implementation-Specific Instruction Set

The G2_LE core instruction set is defined as follows:

- The core provides hardware support for all 32-bit PowerPC instructions.
- The core provides the following two implementation-specific instructions used for software tablesearch operations following TLB misses:
  - Load Data TLB Entry (**tlbld**)
  - Load Instruction TLB Entry (**tlbli**)
- The G2_LE implements the following instruction, which is added to support critical interrupts. This is a supervisor-level, context synchronizing instruction.
  - Return from Critical Interrupt (**rfci**)

2.4 Cache Implementation

The MPC8272 processor core has separate data and instruction caches. The cache implementation is described in the following sections.

2.4.1 PowerPC Cache Model

The PowerPC architecture does not define hardware aspects of cache implementations. For example, some processors, including the MPC8272's processor core, have separate instruction and data caches (Harvard architecture), while others implement a unified cache.

Microprocessors that implement the PowerPC architecture control the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited mode
- Memory coherency

The PowerPC cache management instructions provide a means by which the application programmer can affect the cache contents.

2.4.2 MPC8272 Implementation-Specific Cache Implementation

As shown in Figure 2-1, the caches provide a 64-bit interface to the instruction fetch unit and load/store unit. 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.

Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A27–A31 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 cache blocks are loaded in to the processor core in four beats of 64 bits each. The burst load is performed as critical double word first.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the processor core implements the MEI protocol. These three states, modified, exclusive, and invalid, indicate the state of the cache block as follows:

- 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.
- Invalid—This cache block does not hold valid data.

2.4.2.1 Data Cache

As shown in Figure 2-6, the data cache is configured as 128 sets of four blocks each. Each block consists of 32 bytes, two 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 a cacheable unit. For the MPC8272's processor core, the block size is equivalent to a cache line.

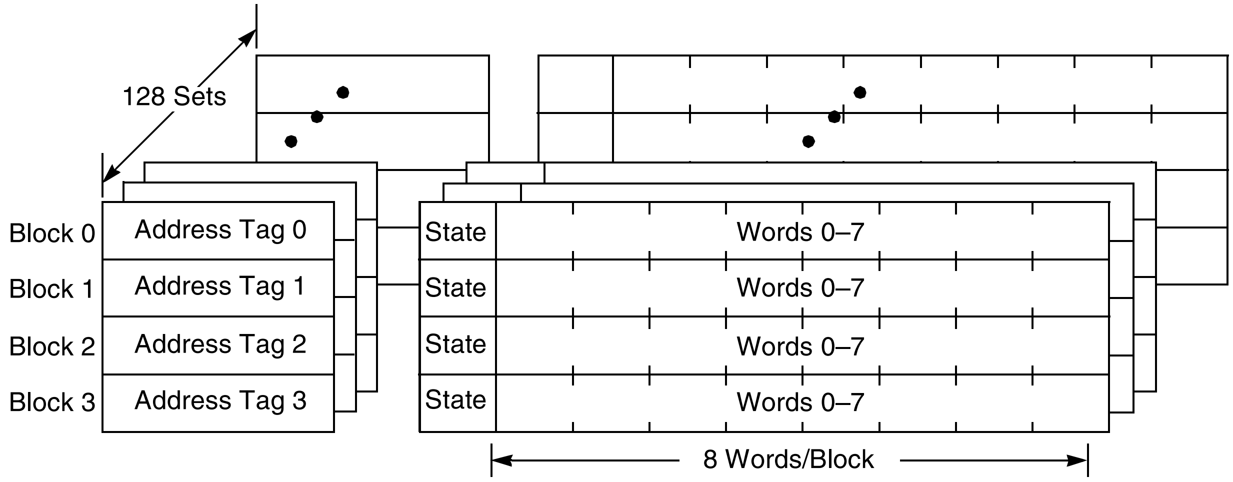


Figure 2-6. Data Cache Organization

Because the processor core data cache tags are single-ported, simultaneous load or 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 which case the snoop is retried and must rearbitrate for access to the cache. Loads or stores that are deferred due to snoop accesses are executed on the clock cycle following the snoop.

Because the caches on the processor core are write-back caches, the predominant type of transaction for most applications is burst-read memory operations, followed by burst-write memory operations, and single-beat (noncacheable or write-through) memory read and write operations. When a cache block is filled with a burst read, the critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

Additionally, there can 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 line in the cache).

Setting HID0[ABE] causes execution of the **dcbf**, **dcbi**, and **dcbst** instructions to be broadcast onto the 60x bus. The value of ABE does not affect **dcbz** instructions, which are always broadcast and snooped. The cache operations are intended primarily for managing on-chip caches. However, the optional broadcast feature is necessary to allow proper management of a system using an external copyback L2 cache.

The address and data buses operate independently to support pipelining and split transactions during memory accesses. The processor core pipelines its own transactions to a depth of one level.

Typically, memory accesses are weakly ordered—sequences of operations, including load/store string and multiple instructions, do not necessarily complete in the order they begin—maximizing the efficiency of the internal bus without sacrificing coherency of the data. The processor core allows pending read operations to precede previous store operations (except when a dependency exists, or in cases where a non-cacheable 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.

2.4.2.2 Instruction Cache

The instruction cache also consists of 128 sets of four blocks, and 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 caused by a cache miss. In the processor core, internal access to the instruction cache is blocked only until the critical load completes.

The processor core supports instruction fetching from other instruction cache lines following the forwarding of the critical first double word of a cache line load operation. The processor core's instruction cache is blocked only until the critical load completes (hits under reloads are allowed). 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 is very similar to the data cache shown in Figure 2-6.

2.4.2.3 Cache Locking

The processor core supports cache locking, which is the ability to prevent some or all of a microprocessor's instruction or data cache from being overwritten. Cache entries can be locked for either an entire cache or for individual ways within the cache. Entire data cache locking is enabled by setting HID0[DLOCK], and entire instruction cache locking is enabled by setting HID0[ILOCK]. For more information, refer to the application note *Cache Locking on the G2 Core* (order number: AN1767). Cache way locking is controlled by the IWLCK and DWLCK bits of HID2.

2.4.2.3.1 Entire Cache Locking

When an entire cache is locked, hits within the cache are supplied in the same manner as hits to an unlocked cache. Any access that misses in the cache is treated as a cache-inhibited access. Cache entries that are invalid at the time of locking will remain invalid and inaccessible until the cache is unlocked. Once the cache has been unlocked, all entries (including invalid entries) are available. Entire cache locking is inefficient if the number of instructions or the size of data to be locked is small compared to the cache size.

2.4.2.3.2 Way Locking

Locking only a portion of the cache is accomplished by locking ways within the cache. Locking always begins with the first way (way0) and is sequential, that is, it is valid to lock ways 0, 1, and 2 but it is not possible to lock just way0 and way2. When using way locking at least one way must be left unlocked. The maximum number of lockable ways is three.

Unlike entire cache locking, invalid entries in a locked way are accessible and available for data placement. As hits to the cache fill invalid entries within a locked way, the entries become valid and locked. This behavior differs from entire cache locking, where nothing is placed in the cache, even if invalid entries exist in the cache. Unlocked ways of the cache behave normally.

2.5 Exception Model

This section describes the PowerPC exception model and implementation-specific details of the MPC8272 core.

2.5.1 PowerPC Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. When exceptions occur, information about the state of the processor is saved to certain registers and the processor begins execution at an address (exception vector) predetermined for each exception. Processing of exceptions occurs in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception—for example, the DSISR identifies instructions that cause a DSI exception. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC architecture requires that exceptions be handled in program order; therefore, although a particular implementation may recognize exception conditions out of order, exceptions are taken in strict order. When an instruction-caused exception 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 exception is taken. Any exceptions caused by those instructions are handled first. Likewise, exceptions that are asynchronous and precise are recognized when they occur, but are not handled until the instruction currently in the completion stage successfully completes execution or generates an exception, and the completed store queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. If, for example, a single instruction encounters multiple exception conditions, those conditions are handled sequentially. After the exception handler handles an exception, the instruction execution continues until the next exception condition is encountered. However, in many cases there is no attempt to re-execute the instruction. This method of recognizing and handling exception conditions sequentially guarantees that exceptions are recoverable.

Exception handlers should save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset or machine check exception or to an instruction-caused exception in the exception handler. SRR0 and SRR1 should also be saved before enabling external interrupts.

The PowerPC architecture supports four types of exceptions:

- Synchronous, precise—These are caused by instructions. All instruction-caused exceptions are handled precisely; that is, the machine state at the time the exception occurs is known and can be completely restored. This means that (excluding the trap and system call exceptions) the address of the faulting instruction is provided to the exception handler and that neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the exception is taken. Once the exception is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the exception handler). When an exception 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 G2 core provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, all enabled floating-point enabled exceptions are always precise on the core).
- Asynchronous, maskable—The external, system management interrupt (SMI), and decrementer interrupts are maskable asynchronous exceptions. When these exceptions occur, their handling is postponed until the next instruction and any exceptions associated with that instruction complete execution. If no instructions are in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0).
- Asynchronous, nonmaskable—There are two nonmaskable asynchronous exceptions: system reset and the machine check exception. These exceptions may not be recoverable, or may provide a limited degree of recoverability. All exceptions report recoverability through MSR[RI].

2.5.2 Implementation-Specific Exception Model

As specified by the PowerPC architecture, all processor core exceptions can be described as either precise or imprecise and either synchronous or asynchronous. Asynchronous exceptions (some of which are maskable) are caused by events external to the processor’s execution. Synchronous exceptions, which are all handled precisely by the processor core, are caused by instructions. The processor core exception classes are shown in Table 2-4.

Table 2-4. Exception Classifications for the Processor Core

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check System reset
Asynchronous, maskable	Precise	External interrupt Decrementer System management interrupt Critical interrupt
Synchronous	Precise	Instruction-caused exceptions

Although exceptions have other characteristics as well, such as whether they are maskable or nonmaskable, the distinctions shown in Table 2-4 define categories of exceptions that the processor core handles uniquely. Note that Table 2-4 includes no synchronous imprecise instructions.

The processor core’s exceptions, and conditions that cause them, are listed in Table 2-5.

Table 2-5. Exceptions and Conditions

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	A system reset is caused by the assertion of either $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$. Note that the reset value of the MSR exception prefix bit (MSR[IP]), described in the *G2 Core Reference Manual*, is determined by the CIP bit in the hard reset configuration word. This is described in Section 5.4.1, “Hard Reset Configuration Word.”
Machine check	00200	A machine check is caused by the assertion of the $\overline{\text{TEA}}$ signal during a data bus transaction, assertion of $\overline{\text{MCP}}$, or an address or data parity error.
DSI	00300	The cause of a DSI exception can be 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. • 4Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared. • 5Set by an **eciwx** or **ecowx** instruction if the access is to an address that is marked as write-through, or execution of a load/store instruction that accesses a direct-store segment. • 6Set for a store operation and cleared for a load operation. • 11Set if **eciwx** or **ecowx** is used and EAR[E] is cleared.
ISI	00400	An ISI exception is 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 exception must be taken to load the PTE (and possibly the page) into memory. • The fetch access is to a direct-store segment (indicated by SRR1[3] set). • 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	An external interrupt is caused when MSR[EE] = 1 and the $\overline{\text{INT}}$ signal is asserted.
Alignment	00600	An alignment exception is caused when the processor core cannot perform a memory access for any of the reasons described below: • The operand of a floating-point load or store is to a direct-store segment. • The operand of a floating-point load or store is not word-aligned. • The operand of a **lmw**, **stmw**, **lwarx**, or **stwcx.** is not word-aligned. • The operand of an elementary, multiple or string load or store crosses a segment boundary with a change to the direct store T bit. • The operand of **dcbz** instruction is in memory that is write-through required or caching inhibited, or **dcbz** is executed in an implementation that has either no data cache or a write-through data cache. • A misaligned **eciwx** or **ecowx** instruction • A multiple or string access with MSR[LE] set The processor core differs from *MPC603e User’s Manual* in that it initiates an alignment exception when it detects a misaligned **eciwx** or **ecowx** instruction and does not initiate an alignment exception when a little-endian access is misaligned.

Table 2-5. Exceptions and Conditions (continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	A program exception is caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction: • Illegal instruction—An illegal instruction program exception 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 processor core), or when execution of an optional instruction not provided in the processor core is attempted (these do not include those optional instructions that are treated as no-ops). • Privileged instruction—A privileged instruction type program exception is generated when the execution of a privileged instruction is attempted and the MSR register user privilege bit, MSR[PR], is set. In the processor core, this exception 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 processors that implement the PowerPC architecture. • Trap—A trap type program exception is generated when any of the conditions specified in a trap instruction is met.
Floating-point unavailable	00800	A floating-point unavailable exception is caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit is cleared (MSR[FP] = 0).
Decrementer	00900	The decrementer exception occurs when the most significant bit of the decrementer (DEC) register transitions from 0 to 1. Must also be enabled with the MSR[EE] bit.
Critical interrupt	00A00	A critical interrupt is caused when MSR[CE] = 1 and the $\overline{\text{CINT}}$ signal is asserted.
Reserved	00B00–00BFF	—
System call	00C00	A system call exception occurs when a System Call (**sc**) instruction is executed.
Trace	00D00	A trace exception is taken when MSR[SE] = 1 or when the currently completing instruction is a branch and MSR[BE] = 1.
Floating-point assist	00E00	Not implemented.
Reserved	00E10–00FFF	—
Instruction translation miss	01000	An instruction translation miss exception is caused when the effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	A data load translation miss exception is caused when the effective address for a data load operation cannot be translated by the DTLB.
Data store translation miss	01200	A data store translation miss exception is caused when the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs, and the changed bit in the PTE must be set due to a data store operation.
Instruction address breakpoint	01300	An instruction address breakpoint exception occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit, and the IABR enable bit (bit 30) is set.

Table 2-5. Exceptions and Conditions (continued)

Exception Type	Vector Offset (hex)	Causing Conditions
System management interrupt	01400	A system management interrupt is caused when MSR[EE] = 1 and the $\overline{\text{SMI}}$ input signal is asserted.
Reserved	01500–02FFF	—

2.6 Memory Management

The following subsections describe the memory management unit (MMU) features of the PowerPC architecture and the G2_LE implementation.

2.6.1 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 exception model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged memory 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.

2.6.2 Implementation-Specific MMU Features

The instruction and data memory management units in the G2_LE 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 G2_LE core rely on the exception 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 maintain address translations for blocks of memory, the G2_LE 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. Adding the HID2[HBE] to the G2_LE enables or disables 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.

As specified by the PowerPC architecture, the page table contains a number of PTEGs. A PTEG contains eight PTEs of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

2.7 Instruction Timing

The G2 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 executable 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 exception, the execution unit reports the exception to the completion/write-back pipeline stage and discontinues instruction execution until the exception is handled. The exception 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 be executing in the FPU concurrently. The FPU pipeline stages are multiply, add, and

round-convert. The LSU has two pipeline stages. The first stage is for effective address calculation and MMU translation, and the second is 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 exception, all following 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 G2 core has five independent execution units, one each for integer instructions, floating-point instructions, branch instructions, load/store instructions, and system register instructions. The IU and the FPU each have dedicated register files for maintaining operands (GPRs and FPRs, respectively), allowing integer and floating-point calculations to occur simultaneously without interference. Integer division performance of the G2 core has been improved, with the **divwu***x* and **divw***x* instructions executing in 20 clock cycles instead of the 37 cycles required in the MPC603e.

The core provides support for single-cycle store and 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.

2.8 Differences Between the MPC8272 G2_LE Embedded Core and the MPC603e

The G2_LE processor core is a derivative of the MPC603e microprocessor design. Some changes have been made and are visible either to a programmer or a system designer. Any software designed around an MPC603e is functional when replaced with the G2_LE except for the specific customer-visible changes listed in Table 2-6.

Software can distinguish between the MPC603e and the G2_LE by reading the processor version register (PVR). The G2_LE processor version number is 0x0081; the processor revision level starts at 0x0100 and is incremented for each revision of the chip.

Table 2-6. Differences Between G2_LE Core and MPC603e

Description	Impact
An additional input interrupt signal, $\overline{\text{CINT}}$, implements a critical interrupt function.	MSR[CE] is allocated for enabling the critical interrupt
A new instruction is implemented for critical interrupt	Return from Critical Interrupt (**rfci**) is implemented to return from these exception handlers
Vector offset for critical interrupt	An exception vector offset of 0x00A00 is defined for critical interrupt
Two new registers are implemented for saving processor state for critical interrupts	CSRR0 and CSRR1 have the same bit assignments as SRR0 and SRR1, respectively.

Table 2-6. Differences Between G2_LE Core and MPC603e (continued)

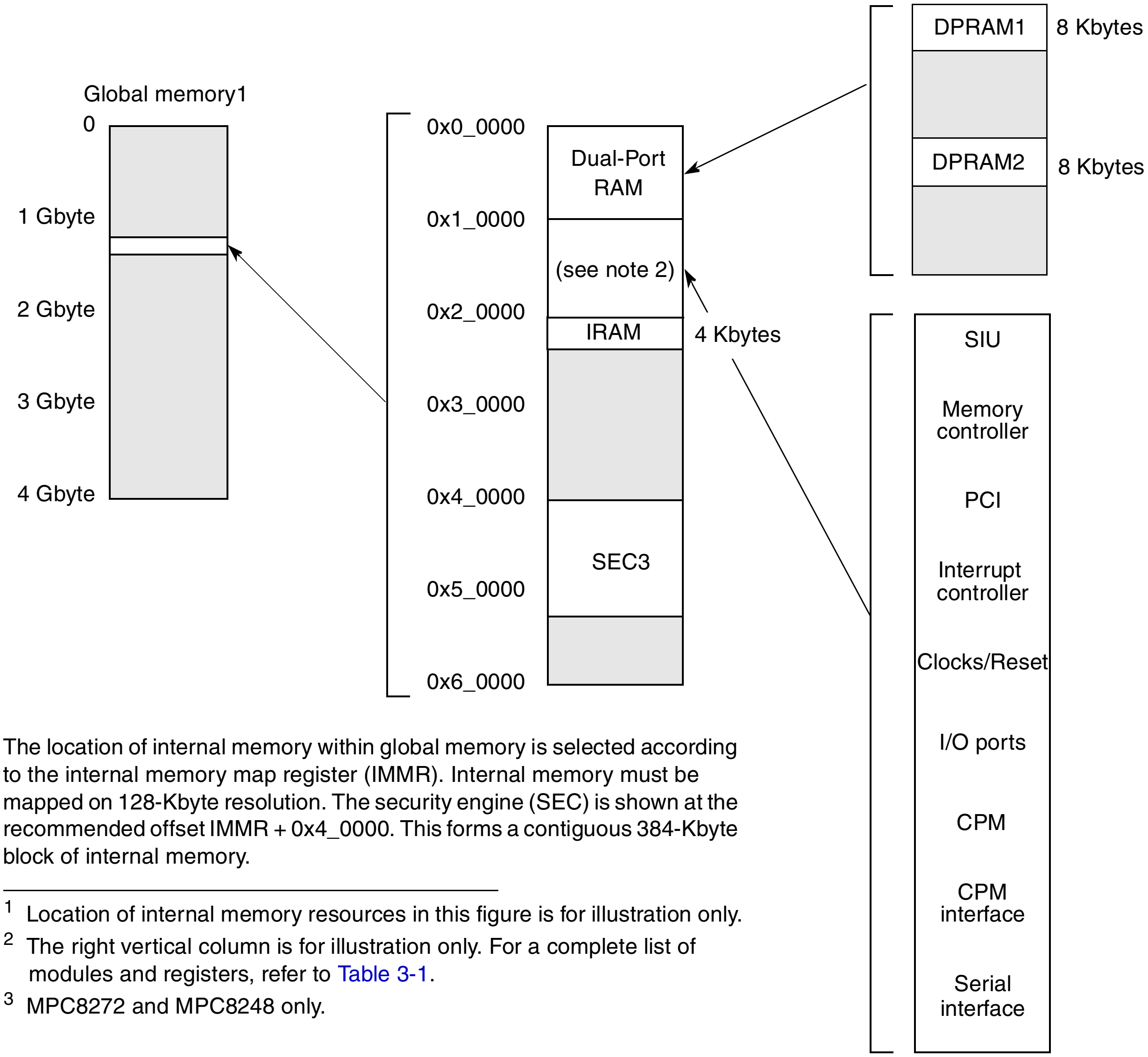
Description	Impact
Supports instruction and data cache way-locking in addition to entire instruction and data cache locking	Implements a cache way locking mechanism for both the instruction and data caches. One to three of the four ways in the cache can be locked with control bits in the HID2 register. See Section 2.3.1.2.3, "Hardware Implementation-Dependent Register 2 (HID2)."
Four additional SPRG registers	The additional SPRGs reduce latencies that may be incurred from saving registers to memory while in an exception handler
One new address breakpoint register IABR2	Instruction address breakpoint exceptions in both the MPC603e and the G2_LE cores use the 0x01300 vector offset
Two new data address breakpoint registers are implemented in the G2_LE	The two new data address breakpoint registers (DABR and DABR2) expand the debug functionality of the breakpoints. The new breakpoint registers are accessible as SPRs with **mtspr** and **mfspr**.
One instruction register and one data breakpoint control register are implemented	IBCR and DBCR are implemented to support the additional debug features. These registers are accessible as SPRs with **mtspr** and **mfspr**.
Vector offset for data address breakpoint exception is 0x00300	Data address breakpoint exception is a DSI exception. The cause of a DSI exception can be determined by the bit settings of DSISR[9]. DAR contains the address of the breakpoint match condition.
One new register is implemented for supporting system level memory map	System memory base address register (MBAR) can be accessed with **mtspr** or **mfspr** using SPR311 in supervisor mode. It can store the present memory base address for the system memory map.
The G2_LE has eight pairs of data and eight pairs of instruction BAT registers	IBAT4–IBAT7 are the four additional pairs of instruction BATs and DBAT4–DBAT7 are the four additional data BATs. HID2[HBE] is added for enabling or disabling the four additional pairs of BAT registers. These BATs are accessible by the **mfspr** and **mtspr** instructions regardless of the setting of HID2[HBE].
Added hardware support for misaligned little endian accesses	Except for strings/multiples, little-endian load/store accesses not on a word boundary generate exceptions under the same circumstances as big-endian accesses.
Removed misalignment support for **eciwx** and **ecowx** instructions	These instructions take an alignment exception if not on a word boundary.
Added ability to broadcast **dcbf**, **dcbi**, and **dcbst** onto the 60x bus	Setting HID0[ABE] enables the new broadcast feature (new in the PID7v-603e). The default is to not broadcast these operations.
Added ability to reflect the value of the M bit onto the 60x bus during instruction translations	Setting HID0[IFEM] enables this feature. The default is to not present the M bit on the bus.
Removed HID0[EICE]	There is no support for ICE pipeline tracking.
Added pin-configurable reset vector	The value of MSR[IP], interrupt prefix, is determined at hard reset by the hardware configuration word.
Addition of speed-for-power functionality	The processor core implements an additional dynamic power management mechanism. HID2[SFP] controls this function. See Section 2.3.1.2.3, "Hardware Implementation-Dependent Register 2 (HID2)."

Chapter 3
Memory Map

The MPC8272's internal memory resources are mapped within a contiguous block of memory. The size of the internal space is 256 Kbytes. The location of this block within the global 4-Gbyte real memory space can be mapped on 128-Kbyte resolution through an implementation-specific special register called the internal memory map register (IMMR). For more information, see Section 4.3.2.5, "Internal Memory Map Register (IMMR)."

In addition, the security engine (SEC) is mapped within a 128-Kbyte block of internal memory. The location of this block must also be mapped on 128-Kbyte resolution. It is recommended to map the SEC block to IMMR + 0x4_0000. This forms a 384-Kbyte contiguous internal memory block. Refer to Section 38.2.1, "SEC Address Base Register."

Figure 3-1 shows the internal memory resources.



The location of internal memory within global memory is selected according to the internal memory map register (IMMR). Internal memory must be mapped on 128-Kbyte resolution. The security engine (SEC) is shown at the recommended offset IMMR + 0x4_0000. This forms a contiguous 384-Kbyte block of internal memory.

[1] Location of internal memory resources in this figure is for illustration only.

[2] The right vertical column is for illustration only. For a complete list of modules and registers, refer to Table 3-1.

[3] MPC8272 and MPC8248 only.

Figure 3-1. Internal Memory

3.1 Internal Memory Map

Table 3-1 defines the internal memory map offset from the address set by the internal memory map register (IMMR).

Table 3-1. Internal Memory Map

Offset	Register	R/W	Size	Reset	Section/Page
CPM Dual-Port RAM					
0x00000–0x01FFF	Dual-port RAM (DPRAM1)	R/W	8 Kbytes	—	13.5/13-15
0x02000–0x07FFF	Reserved	R/W	24 Kbytes	—	—
0x08000–0x09FFF	Dual-port RAM (DPRAM2)	R/W	8 Kbytes	—	13.5/13-15
0x0A000–0x0FFFF	Reserved	—	24 Kbytes	—	—
General SIU					
0x10000	SIU module configuration register (SIUMCR)	R/W	32 bits	See Figure 4-25	4.3.2.4/4-31
0x10004	System protection control register (SYPCR)	R/W	32 bits	0xFFFF_FF07	4.3.2.6/4-34
0x10008	Reserved	—	6 bytes	—	—
0x1000E	Software service register (SWSR)	W	16 bits	Undefined	4.3.2.7/4-35
0x10010–0x10023	Reserved	—	20 bytes	—	—
0x10024	Bus configuration register (BCR)	R/W	32 bits	Reset configuration	4.3.2.1/4-26
0x10028	60x bus arbiter configuration register (PPC_ACR)	R/W	8 bits	See Figure 4-22	4.3.2.2/4-29
0x10029	Reserved	—	24 bits	—	—
0x1002C	60x bus arbitration-level register high (first 8 clients) (PPC_ALRH)	R/W	32 bits	0x0126_3457	4.3.2.3/4-30
0x10030	60x bus arbitration-level register low (next 8 clients) (PPC_ALRL)	R/W	32 bits	0x89AB_CDEF	4.3.2.3/4-30
0x10034	Reserved	—	12 bytes	—	—
0x10040	60x bus transfer error status control register 1 (TESCR1)	R/W	32 bits	0x0000_0000	4.3.2.8/4-36
0x10044	60x bus transfer error status control register 2 (TESCR2)	R/W	32 bits	0x0000_0000	4.3.2.9/4-37
0x10048	Reserved	—	8 bytes	—	—
0x10050	60x bus DMA transfer error address (PDTEA)	R	32 bits	Undefined	18.2.3/18-3
0x10054	60x bus DMA transfer error MSNUM (PDTEM)	R	8 bits	Undefined	18.2.4/18-3

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10055– 0x100FF	Reserved	—	170 bytes	—	—
	Memory Controller				
0x10100	Base register bank 0 (BR0)	R/W	32 bits	See Figure 11-6	11.3.1/11-11
0x10104	Option register bank 0 (OR0)	R/W	32 bits	0xFE00_0EF4	11.3.2/11-13
0x10108	Base register bank 1 (BR1)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x1010C	Option register bank 1 (OR1)	R/W	32 bits	Undefined	11.3.2/11-13
0x10110	Base register bank 2 (BR2)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x10114	Option register bank 2 (OR2)	R/W	32 bits	Undefined	11.3.2/11-13
0x10118	Base register bank 3 (BR3)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x1011C	Option register bank 3 (OR3)	R/W	32 bits	Undefined	11.3.2/11-13
0x10120	Base register bank 4 (BR4)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x10124	Option register bank 4 (OR4)	R/W	32 bits	Undefined	11.3.2/11-13
0x10128	Base register bank 5 (BR5)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x1012C	Option register bank 5 (OR5)	R/W	32 bits	Undefined	11.3.2/11-13
0x10130	Base register bank 6 (BR6)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x10134	Option register bank 6 (OR6)	R/W	32 bits	Undefined	11.3.2/11-13
0x10138	Base register bank 7 (BR7)	R/W	32 bits	0x0000_0000	11.3.1/11-11
0x1013C	Option register bank 7 (OR7)	R/W	32 bits	Undefined	11.3.2/11-13
0x10140–0x 10160	Reserved	—	40 bytes	—	—
0x10168	Memory address register (MAR)	R/W	32 bits	Undefined	11.3.6/11-24
0x1016C	Reserved	—	32 bits	—	—
0x10170	Machine A mode register (MAMR)	R/W	32 bits	0x0004_0000	11.3.4/11-21
0x10174	Machine B mode register (MBMR)	R/W	32 bits	0x0004_0000	
0x10178	Machine C mode register (MCMR)	R/W	32 bits	0x0004_0000	
0x1017C	Reserved	—	48 bits	—	—
0x10184	Memory periodic timer prescaler (MPTPR)	R/W	16 bits	Undefined	11.3.9/11-26
0x10188	Memory data register (MDR)	R/W	32 bits	Undefined	11.3.5/11-23
0x1018C	Reserved	—	32 bits	—	—
0x10190	60x bus SDRAM mode register (PSDMR)	R/W	32 bits	0x0000_0000	11.3.3/11-18
0x10194	Reserved	—	32 bits	—	—
0x10198	60x bus-assigned UPM refresh timer (PURT)	R/W	8 bits	0x00	11.3.7/11-25

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10199	Reserved	—	24 bits	—	—
0x1019C	60x bus-assigned SDRAM refresh timer (PSRT)	R/W	8 bits	0x00	11.3.8/11-25
0x1019D–0x 101A7	Reserved	—	11 bytes	—	—
0x101A8	Internal memory map register (IMMR)	R/W	32 bits	Reset configuration	4.3.2.5/4-33
0x101AC	PCI base register 0 (PCIBR0)	R/W	32 bits	0x0000_0000	4.3.4.1/4-42
0x101B0	PCI base register 1 (PCIBR1)	R/W	32 bits	0x0000_0000	4.3.4.1/4-42
0x101B4	SEC address base register (SECBR)	R/W	32 bits	0x0000_0000	38.2.1/38-10
0x100B8	Reserved	—	32 bits	—	—
0x101BC	SEC mask register (SECMR)	R/W	32 bits	0x0000_0000	38.2.2/38-10
0x101C0–0x101C3	Reserved	—	32 bits	—	—
0x101C4	PCI mask register 0 (PCIMSK0)	R/W	32 bits	0x0000_0000	4.3.4.2/4-43
0x101C8	PCI mask register 1 (PCIMSK1)	R/W	32 bits	0x0000_0000	4.3.4.2/4-43
0x101CC–0x101FF	Reserved	—	52 bytes	—	—
System Integration Timers					
0x10200–0x10 21F	Reserved	—	32 bytes	—	—
0x10220	Time counter status and control register (TMCNTSC)	R/W	16 bits	0x0000	4.3.2.10/4-38
0x10224	Time counter register (TMCNT)	R/W	32 bits	0x0000_0000	4.3.2.11/4-39
0x10228	Reserved	—	32 bits	—	—
0x1022C	Time counter alarm register (TMCNTAL)	R/W	32 bits	0x0000_0000	4.3.2.12/4-39
0x10230–0x1023F	Reserved	—	16 bytes	—	—
0x10240	Periodic interrupt status and control register (PISCR)	R/W	16 bits	0x0000	4.3.3.1/4-40
0x10244	Periodic interrupt count register (PITC)	R/W	32 bits	0x0000_0000	4.3.3.2/4-40
0x10248	Periodic interrupt timer register (PITR)	R	32 bits	0x0000_0000	4.3.3.3/4-41
0x1024C–0x102A8	Reserved	—	92 bytes	—	—
0x102AA–0x1042F	Reserved	—	372 bytes	—	—
PCI					
0x10430	Outbound interrupt status register (OMISR)	R/W	32 bits	0x0000_0000	9.12.3.4.3/9-80

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10434	Outbound interrupt mask register (OMIMR)	R/W	32 bits	0x0000_0000	9.12.3.4.4/9-81
0x10440	Inbound FIFO queue port register (IFQPR)	R/W	32 bits	0x0000_0000	9.12.3.4.1/9-79
0x10444	Outbound FIFO queue port register (OFQPR)	R/W	32 bits	0x0000_0000	9.12.3.4.2/9-80
0x10450	Inbound message register 0 (IMR0)	R/W	32 bits	Undefined	9.12.1.1/9-68
0x10454	Inbound message register 1 (IMR1)	R/W	32 bits	Undefined	9.12.1.1/9-68
0x10458	Outbound message register 0 (OMR0)	R/W	32 bits	Undefined	9.12.1.2/9-69
0x1045C	Outbound message register 1 (OMR1)	R/W	32 bits	Undefined	9.12.1.2/9-69
0x10460	Outbound doorbell register (ODR)	R/W	32 bits	0x0000_0000	9.12.2.1/9-70
0x10468	Inbound doorbell register (IDR)	R/W	32 bits	0x0000_0000	9.12.2.2/9-70
0x10480	Inbound message interrupt status register (IMISR)	R/W	32 bits	0x0000_0000	9.12.3.4.5/9-82
0x10484	Inbound message interrupt mask register (IMIMR)	R/W	32 bits	0x0000_0000	9.12.3.4.6/9-84
0x104A0	Inbound free_FIFO head pointer register (IFHPR)	R/W	32 bits	0x0000_0000	9.12.3.2.1/9-73
0x104A8	Inbound free_FIFO tail pointer register (IFTPR)	R/W	32 bits	0x0000_0000	9.12.3.2.1/9-73
0x104B0	Inbound post_FIFO head pointer register (IPHPR)	R/W	32 bits	0x0000_0000	9.12.3.2.2/9-74
0x104B8	Inbound post_FIFO tail pointer register (IPTPR)	R/W	32 bits	0x0000_0000	9.12.3.2.2/9-74
0x104C0	Outbound free_FIFO head pointer register (OFHPR)	R/W	32 bits	0x0000_0000	9.12.3.3.1/9-76
0x104C8	Outbound free_FIFO tail pointer register (OFTPR)	R/W	32 bits	0x0000_0000	9.12.3.3.1/9-76
0x104D0	Outbound post_FIFO head pointer register (OPHPR)	R/W	32 bits	0x0000_0000	9.12.3.3.2/9-77
0x104D8	Outbound post_FIFO tail pointer register (OPTPR)	R/W	32 bits	0x0000_0000	9.12.3.3.2/9-77
0x104E4	Message unit control register (MUCR)	R/W	32 bits	0x0000_0002	9.12.3.4.7/9-85
0x104F0	Queue base address register (QBAR)	R/W	32 bits	0x0000_0000	9.12.3.4.8/9-86
0x10500	DMA 0 mode register (DMAMR0)	R/W	32 bits	0x0000_0000	9.13.1.6.1/9-90
0x10504	DMA 0 status register (DMASR0)	R/W	32 bits	0x0000_0000	9.13.1.6.2/9-92
0x10508	DMA 0 current descriptor address register (DMACDAR0)	R/W	32 bits	0x0000_0000	9.13.1.6.3/9-93
0x10510	DMA 0 source address register (DMASAR0)	R/W	32 bits	0x0000_0000	9.13.1.6.4/9-94
0x10518	DMA 0 destination address register (DMADAR0)	R/W	32 bits	0x0000_0000	9.13.1.6.5/9-95
0x10520	DMA 0 byte count register (DMABCR0)	R/W	32 bits	0x0000_0000	9.13.1.6.6/9-95
0x10524	DMA 0 next descriptor address register (DMANDAR0)	R/W	32 bits	0x0000_0000	9.13.1.6.7/9-96
0x10580	DMA 1 mode register (DMAMR1)	R/W	32 bits	0x0000_0000	9.13.1.6.1/9-90
0x10584	DMA 1 status register (DMASR1)	R/W	32 bits	0x0000_0000	9.13.1.6.2/9-92
0x10588	DMA 1 current descriptor address register (DMACDAR1)	R/W	32 bits	0x0000_0000	9.13.1.6.3/9-93

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10590	DMA 1 source address register (DMASAR1)	R/W	32 bits	0x0000_0000	9.13.1.6.4/9-94
0x10598	DMA 1 destination address register (DMADAR1)	R/W	32 bits	0x0000_0000	9.13.1.6.5/9-95
0x105A0	DMA 1 byte count register (DMABCR1)	R/W	32 bits	0x0000_0000	9.13.1.6.6/9-95
0x105A4	DMA 1 next descriptor address register (DMANDAR1)	R/W	32 bits	0x0000_0000	9.13.1.6.7/9-96
0x10600	DMA 2 mode register (DMAMR2)	R/W	32 bits	0x0000_0000	9.13.1.6.1/9-90
0x10604	DMA 2 status register (DMASR2)	R/W	32 bits	0x0000_0000	9.13.1.6.2/9-92
0x10608	DMA 2 current descriptor address register (DMACDAR2)	R/W	32 bits	0x0000_0000	9.13.1.6.3/9-93
0x10610	DMA 2 source address register (DMASAR2)	R/W	32 bits	0x0000_0000	9.13.1.6.4/9-94
0x10618	DMA 2 destination address register (DAR2)	R/W	32 bits	0x0000_0000	9.13.1.6.5/9-95
0x10620	DMA 2 byte count register (DMABCR2)	R/W	32 bits	0x0000_0000	9.13.1.6.6/9-95
0x10624	DMA 2 next descriptor address register (DMANDAR2)	R/W	32 bits	0x0000_0000	9.13.1.6.7/9-96
0x10680	DMA 3 mode register (DMAMR3)	R/W	32 bits	0x0000_0000	9.13.1.6.1/9-90
0x10684	DMA 3 status register (DMASR3)	R/W	32 bits	0x0000_0000	9.13.1.6.2/9-92
0x10688	DMA 3 current descriptor address register (DMACDAR3)	R/W	32 bits	0x0000_0000	9.13.1.6.3/9-93
0x10690	DMA 3 source address register (DMASAR3)	R/W	32 bits	0x0000_0000	9.13.1.6.4/9-94
0x10698	DMA 3 destination address register (DMADAR3)	R/W	32 bits	0x0000_0000	9.13.1.6.5/9-95
0x106A0	DMA 3 byte count register (DMABCR3)	R/W	32 bits	0x0000_0000	9.13.1.6.6/9-95
0x106A4	DMA 3 next descriptor address register (DMANDAR3)	R/W	32 bits	0x0000_0000	9.13.1.6.7/9-96
0x10800	PCI outbound translation address register 0 (POTAR0)	R/W	32 bits	0x0000_0000	9.11.1.3/9-31
0x10808	PCI outbound base address register 0 (POBAR0)	R/W	32 bits	0x0000_0000	9.11.1.4/9-32
0x10810	PCI outbound comparison mask register 0 (POCMR0)	R/W	32 bits	0x0000_0000	9.11.1.5/9-33
0x10818	PCI outbound translation address register 1 (POTAR1)	R/W	32 bits	0x0000_0000	9.11.1.3/9-31
0x10820	PCI outbound base address register 1 (POBAR1)	R/W	32 bits	0x0000_0000	9.11.1.4/9-32
0x10828	PCI outbound comparison mask register 1 (POCMR1)	R/W	32 bits	0x0000_0000	9.11.1.5/9-33
0x10830	PCI outbound translation address register 2 (POTAR2)	R/W	32 bits	0x0000_0000	9.11.1.3/9-31
0x10838	PCI outbound base address register 2 (POBAR2)	R/W	32 bits	0x0000_0000	9.11.1.4/9-32
0x10840	PCI outbound comparison mask register 2 (POCMR2)	R/W	32 bits	0x0000_0000	9.11.1.5/9-33
0x10878	Discard timer control register (PTCR)	R/W	32 bits	0x0000_0000	9.11.1.6/9-34
0x1087C	General purpose control register (GPCR)	R/W	32 bits	0x0000_0000	9.11.1.7/9-34
0x10880	PCI general control register (PCI_GCR)	R/W	32 bits	0x0000_0000	9.11.1.8/9-36
0x10884	Error status register (ESR)	R/W	32 bits	0x0000_0000	9.11.1.9/9-36

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10888	Error mask register (EMR)	R/W	32 bits	0x0000_0FFF	9.11.1.10/9-38
0x1088C	Error control register (ECR)	R/W	32 bits	0x0000_00FF	9.11.1.11/9-39
0x10890	PCI error address capture register (PCI_EACR)	R/W	32 bits	0x0000_0000	9.11.1.12/9-40
0x10898	PCI error data capture register (PCI_EDCR)	R/W	32 bits	0x0000_0000	9.11.1.13/9-41
0x108A0	PCI error control capture register (PCI_ECCR)	R/W	32 bits	0x0000_0000	9.11.1.14/9-41
0x108D0	PCI inbound translation address register 1 (PITAR1)	R/W	32 bits	0x0000_0000	9.11.1.15/9-43
0x108D8	PCI inbound base address register 1 (PIBAR1)	R/W	32 bits	0x0000_0000	9.11.1.16/9-43
0x108E0	PCI inbound comparison mask register 1 (PICMR1)	R/W	32 bits	0x0000_0000	9.11.1.17/9-44
0x108E8	PCI inbound translation address register 0 (PITAR0)	R/W	32 bits	0x0000_0000	9.11.1.15/9-43
0x108F0	PCI inbound base address register 0 (PIBAR0)	R/W	32 bits	0x0000_0000	9.11.1.16/9-43
0x108F8	PCI inbound comparison mask register 0 (PICMR0)	R/W	32 bits	0x0000_0000	9.11.1.17/9-44
0x10900	PCI CFG_ADDR	R/W	32 bits	Undefined	9.9.1.4.4/9-15
0x10904	PCI CFG_DATA	R/W	32 bits	0x0000_0000	9.9.1.4.4/9-15
0x10908	PCI INT_ACK	R/W	32 bits	Undefined	9.9.1.4.7/9-18
Interrupt Controller					
0x10C00	SIU interrupt configuration register (SICR)	R/W	16 bits	0x0000	4.3.1.1/4-17
0x10C02	Reserved	—	16 bits	—	—
0x10C04	SIU interrupt vector register (SIVEC)	R/W	32 bits	0x0000_0000	4.3.1.6/4-24
0x10C08	SIU interrupt pending register (high) (SIPNR_H)	R/W	32 bits	Undefined	4.3.1.4/4-21
0x10C0C	SIU interrupt pending register (low) (SIPNR_L)	R/W	32 bits	0x0000_0000	4.3.1.4/4-21
0x10C10	SIU interrupt priority register (SIPRR)	R/W	32 bits	0x0530_9770	4.3.1.2/4-18
0x10C14	CPM interrupt priority register (high) (SCPRR_H)	R/W	32 bits	0x0530_9770	4.3.1.3/4-19
0x10C18	CPM interrupt priority register (low) (SCPRR_L)	R/W	32 bits	0x0530_9770	4.3.1.3/4-19
0x10C1C	SIU interrupt mask register (high) (SIMR_H)	R/W	32 bits	0x0000_0000	4.3.1.5/4-22
0x10C20	SIU interrupt mask register (low) (SIMR_L)	R/W	32 bits	0x0000_0000	4.3.1.5/4-22
0x10C24	SIU external interrupt control register (SIEXR)	R/W	32 bits	0x0000_0000	4.3.1.7/4-25
0x10C28– 0x10C7F	Reserved	—	88 bytes	—	—
Clocks and Reset					
0x10C80	System clock control register (SCCR)	R/W	32 bits	See Table 10-2	10.4/10-6
0x10C88	System clock mode register (SCMR)	R	32 bits	See Table 10-3	10.5/10-7
0x10C90	Reset status register (RSR)	R/W	32 bits	0x0000_0003	5.2/5-4
0x10C94	Reset mode register (RMR)	R/W	32 bits	0x0000_0000	5.3/5-5

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10C98–0x10CFF	Reserved	—	104 bytes	—	—
Input/Output Port					
0x10D00	Port A data direction register (PDIRA)	R/W	32 bits	0x0000_0000	37.2.2.2/37-5
0x10D04	Port A pin assignment register (PPARA)	R/W	32 bits	0x0000_0000	37.2.4/37-9
0x10D08	Port A special options register (PSORA)	R/W	32 bits	0x0000_0000	37.2.5/37-13
0x10D0C	Port A open drain register (PODRA)	R/W	32 bits	0x0000_0000	37.2.1/37-1
0x10D10	Port A data register (PDATA)	R/W	32 bits	0x0000_0000	37.2.2/37-4
0x10D14–0x10D1F	Reserved	—	12 bytes	—	—
0x10D20	Port B data direction register (PDIRB)	R/W	32 bits	0x0000_0000	37.2.2.2/37-5
0x10D24	Port B pin assignment register (PPARB)	R/W	32 bits	0x0000_0000	37.2.4/37-9
0x10D28	Port B special operation register (PSORB)	R/W	32 bits	0x0000_0000	37.2.5/37-13
0x10D2C	Port B open drain register (PODRB)	R/W	32 bits	0x0000_0000	37.2.1/37-1
0x10D30	Port B data register (PDATB)	R/W	32 bits	0x0000_0000	37.2.2/37-4
0x10D34–0x10D3F	Reserved	—	12 bytes	—	—
0x10D40	Port C data direction register (PDIRC)	R/W	32 bits	0x0000_0000	37.2.2.2/37-5
0x10D44	Port C pin assignment register (PPARC)	R/W	32 bits	0x0000_0000	37.2.4/37-9
0x10D48	Port C special operation register (PSORC)	R/W	32 bits	0x0000_0000	37.2.5/37-13
0x10D4C	Port C open drain register (PODRC)	R/W	32 bits	0x0000_0000	37.2.1/37-1
0x10D50	Port C data register (PDATC)	R/W	32 bits	0x0000_0000	37.2.2/37-4
0x10D54–0x10D5F	Reserved	—	12 bytes	—	—
0x10D60	Port D data direction register (PDIRD)	R/W	32 bits	0x0000_0000	37.2.2.2/37-5
0x10D64	Port D pin assignment register (PPARD)	R/W	32 bits	0x0000_0000	37.2.4/37-9
0x10D68	Port D special operation register (PSORD)	R/W	32 bits	0x0000_0000	37.2.5/37-13
0x10D6C	Port D open drain register (PODRD)	R/W	32 bits	0x0000_0000	37.2.1/37-1
0x10D70	Port D data register (PDATD)	R/W	32 bits	0x0000_0000	37.2.2/37-4
CPM Timers					
0x10D80	Timer 1 and timer 2 global configuration register (TGCR1)	R/W	8 bits	0x00	17.2.2/17-3
0x10D81	Reserved	—	24 bits	—	—
0x10D84	Timer 3 and timer 4 global configuration register (TGCR2)	R/W	8 bits	0x00	17.2.2/17-3

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x10D85– 0x10D8F	Reserved	—	11 bytes	—	—
0x10D90	Timer 1 mode register (TMR1)	R/W	16 bits	0x0000	17.2.3/17-5
0x10D92	Timer 2 mode register (TMR2)	R/W	16 bits	0x0000	17.2.3/17-5
0x10D94	Timer 1 reference register (TRR1)	R/W	16 bits	0x0000	17.2.4/17-6
0x10D96	Timer 2 reference register (TRR2)	R/W	16 bits	0x0000	17.2.4/17-6
0x10D98	Timer 1 capture register (TCR1)	R/W	16 bits	0x0000	17.2.5/17-7
0x10D9A	Timer 2 capture register (TCR2)	R/W	16 bits	0x0000	17.2.5/17-7
0x10D9C	Timer 1 counter (TCN1)	R/W	16 bits	0x0000	17.2.6/17-7
0x10D9E	Timer 2 counter (TCN2)	R/W	16 bits	0x0000	17.2.6/17-7
0x10DA0	Timer 3 mode register (TMR3)	R/W	16 bits	0x0000	17.2.3/17-5
0x10DA2	Timer 4 mode register (TMR4)	R/W	16 bits	0x0000	17.2.3/17-5
0x10DA4	Timer 3 reference register (TRR3)	R/W	16 bits	0x0000	17.2.4/17-6
0x10DA6	Timer 4 reference register (TRR4)	R/W	16 bits	0x0000	17.2.4/17-6
0x10DA8	Timer 3 capture register (TCR3)	R/W	16 bits	0x0000	17.2.5/17-7
0x10DAA	Timer 4 capture register (TCR4)	R/W	16 bits	0x0000	17.2.5/17-7
0x10DAC	Timer 3 counter (TCN3)	R/W	16 bits	0x0000	17.2.6/17-7
0x10DAE	Timer 4 counter (TCN4)	R/W	16 bits	0x0000	17.2.6/17-7
0x10DB0	Timer 1 event register (TER1)	R/W	16 bits	0x0000	17.2.7/17-7
0x10DB2	Timer 2 event register (TER2)	R/W	16 bits	0x0000	17.2.7/17-7
0x10DB4	Timer 3 event register (TER3)	R/W	16 bits	0x0000	17.2.7/17-7
0x10DB6	Timer 4 event register (TER4)	R/W	16 bits	0x0000	17.2.7/17-7
0x10DB8– 0x11017	Reserved	—	608 bytes	—	—
SDMA–General					
0x11018	SDMA status register (SDSR)	R/W	8 bits	0x00	18.2.1/18-3
0x11019	Reserved	—	24 bits	—	—
0x1101C	SDMA mask register (SDMR)	R/W	8 bits	0x00	18.2.2/18-3
0x1101D	Reserved	—	24 bits	—	—
IDMA					
0x11020	Reserved	—	64 bits	—	—
0x11028	IDMA 2 event register (IDSR2)	R/W	8 bits	0x00	18.8.4/18-22
0x11029	Reserved	—	24 bits	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x1102C	IDMA 2 mask register (IDMR2)	R/W	8 bits	0x00	18.8.4/18-22
0x1102D	Reserved	—	24 bits	—	—
0x11030	IDMA 3 event register (IDSR3)	R/W	8 bits	0x00	18.8.4/18-22
0x11031	Reserved	—	24 bits	—	—
0x11034	IDMA 3 mask register (IDMR3)	R/W	8 bits	0x00	18.8.4/18-22
0x11035–0x112FF	Reserved	—	715 bytes	—	—
FCC1					
0x11300	FCC1 general mode register (GFMR1)	R/W	32 bits	0x0000_0000	29.2/29-3
0x11304	FCC1 protocol-specific mode register (FPSMR1)	R/W	32 bits	0x0000_0000	30.13.3/30-84 (ATM) 32.18.2/32-20 (Ethernet) 33.6/33-7 (HDLC)
0x11308	FCC1 transmit on demand register (FTODR1)	R/W	16 bits	0x0000	29.5/29-9
0x1130A	Reserved	—	16 bits	—	—
0x1130C	FCC1 data synchronization register (FDSR1)	R/W	16 bits	0x7E7E	29.4/29-8
0x1130E	Reserved	—	16 bits	—	—
0x11310	FCC1 event register (FCCE1)	R/W	16 bits	0x0000_0000	30.13.4/30-86 (ATM) 32.18.3/32-22 (Ethernet) 33.9/33-14 (HDLC)
0x11312	Reserved	—	16 bits	—	—
0x11314	FCC1 mask register (FCCM1)	R/W	16 bits	0x0000_0000	30.13.4/30-86 (ATM) 32.18.3/32-22 (Ethernet) 33.9/33-14 (HDLC)
0x11316	Reserved	—	16 bits	—	—
0x11318	FCC1 status register (FCCS1)	R	16 bits	0x00	33.10/33-16 (HDLC)
0x11319	Reserved	—	24 bits	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x1131C	FCC1 transmit internal rate registers for PHY0 (FTIRR1_PHY0)	R/W	8 bits	0x00	30.13.5/30-87 (ATM)
0x1131D	FCC1 transmit internal rate registers for PHY1 (FTIRR1_PHY1)	R/W	8 bits	0x00	
0x1131E	FCC1 transmit internal rate registers for PHY2 (FTIRR1_PHY2)	R/W	8 bits	0x00	
0x1131F	FCC1 transmit internal rate registers for PHY3 (FTIRR1_PHY3)	R/W	8 bits	0x00	
FCC2					
0x11320	FCC2 general mode register (GFMR2)	R/W	32 bits	0x0000_0000	29.2/29-3
0x11324	FCC2 protocol-specific mode register (FPSMR2)	R/W	32 bits	0x0000_0000	32.18.2/32-20 (Ethernet) 33.6/33-7 (HDLC)
0x11328	FCC2 transmit on-demand register (FTODR2)	R/W	16 bits	0x0000	29.5/29-9
0x1132A	Reserved	—	16 bits	—	—
0x1132C	FCC2 data synchronization register (FDSR2)	R/W	16 bits	0x7E7E	29.4/29-8
0x1132E	Reserved	—	16 bits	—	—
0x11330	FCC2 event register (FCCE2)	R/W	16 bits	0x0000	32.18.3/32-22 (Ethernet) 33.9/33-14 (HDLC)
0x11332	Reserved	—	16 bits	—	—
0x11334	FCC2 mask register (FCCM2)	R/W	16 bits	0x0000	32.18.3/32-22 (Ethernet) 33.9/33-14 (HDLC)
0x11336	Reserved	—	16 bits	—	—
0x11338	FCC2 status register (FCCS2)	R	16 bits	0x00	33.10/33-16 (HDLC)
0x1133A - 0x1137F	Reserved	—	70 bytes	—	—
FCC1 Extended Registers					
0x11380	FCC1 Internal Rate Port Enable Register (FIRPER1)	—	32 bit	—	30.14.3/30-90
0x11384	FCC1 Internal Rate Event Register (FIRER1)	—	32 bit	—	30.14.4/30-91
0x11388	FCC1 Internal Rate Select Register High (FIRSR1_HI)	—	32 bit	—	30.14.5/30-91
0x1138C	FCC1 Internal Rate Select Register Low (FIRSR1_LO)	—	32 bit	—	30.14.5/30-91
0x11390	FCC1 General Extended Mode Register (GFEMR1)	—	8 bit	—	29.2.2/29-7

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x11391– 0x113AF	Reserved	—	47 bytes	—	—
FCC2 Extended Registers					
0x113B0	FCC2 general expansion mode register (GFEMR2)	R/W	8 bits	—	29.2.2/29-7
0x113B1– 0x115DF	Reserved	—	559 bytes	—	—
BRGs 5–8					
0x115F0	BRG5 configuration register (BRGC5)	R/W	32 bits	0x0000_0000	16.1/16-2
0x115F4	BRG6 configuration register (BRGC6)	R/W	32 bits	0x0000_0000	
0x1115F8	BRG7 configuration register (BRGC7)	R/W	32 bits	0x0000_0000	
0x115FC	BRG8 configuration register (BRGC8)	R/W	32 bits	0x0000_0000	
0x11600– 0x1185F	Reserved	—	608 bytes	—	—
I^{2}C					
0x11860	I^{2}C mode register (I2MOD)	R/W	8 bits	0x00	36.4.1/36-6
0x11861	Reserved	—	24 bits	—	—
0x11864	I^{2}C address register (I2ADD)	R/W	8 bits	0x00	36.4.2/36-6
0x11865	Reserved	—	24 bits	—	—
0x11868	I^{2}C BRG register (I2BRG)	R/W	8 bits	0x00	36.4.3/36-7
0x11869	Reserved	—	24 bits	—	—
0x1186C	I^{2}C command register (I2COM)	R/W	8 bits	0x00	36.4.5/36-8
0x1186D	Reserved	—	24 bits	—	—
0x11870	I^{2}C event register (I2CER)	R/W	8 bits	0x00	36.4.4/36-7
0x11871	Reserved	—	24 bits	—	—
0x11874	I^{2}C mask register (I2CMR)	R/W	8 bits	0x00	36.4.4/36-7
0x11875– 0x119BF	Reserved	—	315 bytes	—	—
Communications Processor					
0x119C0	Communications processor command register (CPCR)	R/W	32 bits	0x0000_0000	13.4.1/13-11
0x119C4	CP configuration register (RCCR)	R/W	32 bits	0x0000_0000	13.3.6/13-7
0x119C8– 0x119D5	Reserved	—	12 bytes	—	—
0x119D6	CP timers event register (RTER)	R/W	16 bits	0x0000_0000	13.6.4/13-21
0x119DA	CP timers mask register (RTMR)	R/W	16 bits	0x0000_0000	

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x119DC	CP time-stamp timer control register (RTSCR)	—	16 bits	0x0000	13.3.7/13-9
0x119DE	Reserved	R/W	16 bits	—	—
0x119E0	CP time-stamp register (RTSR)	R/W	32 bits	0x0000	13.3.8/13-10
BRGs 1–4					
0x119F0	BRG1 configuration register (BRGC1)	R/W	32 bits	0x0000_0000	16.1/16-2
0x119F4	BRG2 configuration register (BRGC2)	R/W	32 bits	0x0000_0000	
0x119F8	BRG3 configuration register (BRGC3)	R/W	32 bits	0x0000_0000	
0x119FC	BRG4 configuration register (BRGC4)	R/W	32 bits	0x0000_0000	
SCC1					
0x11A00	SCC1 general mode register (GSMR_L1)	R/W	32 bits	0x0000_0000	19.1.1/19-3
0x11A04	SCC1 general mode register (GSMR_H1)	R/W	32 bits	0x0000_0000	
0x11A08	SCC1 protocol-specific mode register (PSMR1)	R/W	16 bits	0x0000	19.1.2/19-9 20.16/20-12 (UART) 21.8/21-7 (HDLC) 22.11/22-10 (BISYNC) 23.9/23-8 (Transparent) 24.17/24-14 (Ethernet)
0x11A0A	Reserved	—	16 bits	—	—
0x11A0C	SCC1 transmit-on-demand register (TODR1)	R/W	16 bits	0x0000	19.1.4/19-10
0x11A0E	SCC1 data synchronization register (DSR1)	R/W	16 bits	0x7E7E	19.1.3/19-9
0x11A10	SCC1 event register (SCCE1)	R/W	16 bits	0x0000	20.19/20-19 (UART) 21.11/21-12 (HDLC) 22.14/22-15 (BISYNC) 23.12/23-11 (Transparent) 24.20/24-20 (Ethernet)
0x11A14	SCC1 mask register (SCCM1)	R/W	16 bits	0x0000	
0x11A16	Reserved	—	8 bits	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x11A17	SCC1 status register (SCCS1)	R/W	8 bits	0x00	20.20/20-21 (UART) 21.12/21-14 (HDLC) 22.15/22-16 (BISYNC) 23.13/23-12 (Transparent)
0x11A18–0x11A3F	Reserved	—	40 bytes	—	—
SCC3					
0x11A40	SCC3 general mode register (GSMR_L3)	R/W	32 bits	0x0000_0000	19.1.1/19-3
0x11A44	SCC3 general mode register (GSMR_H3)	R/W	32 bits	0x0000_0000	
0x11A48	SCC3 protocol-specific mode register (PSMR3)	R/W	16 bits	0x0000	19.1.2/19-9 20.16/20-12 (UART) 21.8/21-7 (HDLC) 22.11/22-10 (BISYNC) 23.9/23-8 (Transparent) 24.17/24-14 (Ethernet)
0x11A4A	Reserved	—	16 bits	—	—
0x11A4C	SCC3 transmit on demand register (TODR3)	R/W	16 bits	0x0000	19.1.4/19-10
0x11A4E	SCC3 data synchronization register (DSR3)	R/W	16 bits	0x7E7E	19.1.3/19-9
0x11A50	SCC3 event register (SCCE3)	R/W	16 bits	0x0000	20.19/20-19 (UART) 21.11/21-12 (HDLC) 22.14/22-15 (BISYNC) 23.12/23-11 (Transparent) 24.20/24-20 (Ethernet)
0x11A54	SCC3 mask register (SCCM3)	R/W	16 bits	0x0000	
0x11A56	Reserved	—	8 bits	—	—
0x11A57	SCC3 status register (SCCS3)	R/W	8 bits	0x00	20.20/20-21 (UART) 21.12/21-14 (HDLC) 22.15/22-16 (BISYNC) 23.13/23-12 (Transparent)

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x11A58– 0x11A5F	Reserved	—	8 bytes	—	—
	SCC4				
0x11A60	SCC4 general mode register (GSMR_L4)	R/W	32 bits	0x0000_0000	19.1.1/19-3
0x11A64	SCC4 general mode register (GSMR_H4)	R/W	32 bits	0x0000_0000	
0x11A68	SCC4 protocol-specific mode register (PSMR4)	R/W	16 bits	0x0000	19.1.2/19-9 20.16/20-12 (UART) 21.8/21-7 (HDLC) 22.11/22-10 (BISYNC) 23.9/23-8 (Transparent) 24.17/24-14 (Ethernet)
0x11A6A	Reserved	—	16 bits	—	—
0x11A6C	SCC4 transmit on-demand register (TODR4)	R/W	16 bits	0x0000	19.1.4/19-10
0x11A6E	SCC4 data synchronization register (DSR4)	R/W	16 bits	0x7E7E	19.1.3/19-9
0x11A70	SCC4 event register (SCCE4)	R/W	16 bits	0x0000	20.19/20-19 (UART) 21.11/21-12 (HDLC) 22.14/22-15 (BISYNC) 23.12/23-11 (Transparent) 24.20/24-20 (Ethernet)
0x11A74	SCC4 mask register (SCCM4)	R/W	16 bits	0x0000	
0x11A77	SCC4 status register (SCCS4)	—	8 bits	0x00	20.20/20-21 (UART) 21.12/21-14 (HDLC) 22.15/22-16 (BISYNC) 23.13/23-12 (Transparent)
0x11A78– 0x11A7F	Reserved	—	8 bytes	—	—
	SMC1				
0x11A82	SMC1 mode register (SMCMR1)	R/W	16 bits	0x0000	28.2.1/28-2
0x11A84	Reserved	—	16 bits	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x11A86	SMC1 event register (SMCE1)	R/W	8 bits	0x00	28.3.11/28-18 (UART) 28.4.10/28-28 (Transparent) 28.5.9/28-34 (GCI)
0x11A87	Reserved	—	24 bits	—	
0x11A8A	SMC1 mask register (SMCM1)	R/W	8 bits	0x00	
0x11A8B–0x11A91	Reserved	—	7 bytes	—	—
SMC2					
0x11A92	SMC2 mode register (SMCMR2)	R/W	16 bits	0x0000	28.2.1/28-2
0x11A94	Reserved	—	16 bits	—	—
0x11A96	SMC2 event register (SMCE2)	R/W	8 bits	0x00	28.3.11/28-18 (UART) 28.4.10/28-28 (Transparent) 28.5.9/28-34 (GCI)
0x11A97	Reserved	—	24 bits	—	
0x11A9A	SMC2 mask register (SMCM2)	R/W	8 bits	0x00	
0x11A9B–0x11A9F	Reserved	—	5 bytes	—	—
SPI					
0x11AA0	SPI mode register (SPMODE)	R/W	16 bits	0x0000	35.4.1/35-6
0x11AA2	Reserved	—	4 bytes	—	—
0x11AA6	SPI event register (SPIE)	R/W	8 bits	0x00	35.4.2/35-9
0x11AA7	Reserved	—	24 bits	—	—
0x11AAA	SPI mask register (SPIM)	R/W	8 bits	0x00	35.4.2/35-9
0x11AAB	Reserved	—	24 bits	—	—
0x11AAD	SPI command register (SPCOM)	W	8 bits	0x00	35.4.3/35-10
0x11AAE–0x11AFF	Reserved	—	82 bytes	—	—
CPM Mux					
0x11B00	Reserved	R/W	16 bits	—	—
0x11B02	CPM mux SI2 clock route register (CMXSI2CR)	R/W	8 bits	0x00	15.4.2/15-6
0x11B03	Reserved	—	8 bits	—	—
0x11B04	CPM mux FCC clock route register (CMXFCR)	R/W	32 bits	0x0000_0000	15.4.3/15-7
0x11B08	CPM mux SCC clock route register (CMXSCR)	R/W	32 bits	0x0000_0000	15.4.4/15-9
0x11B0C	CPM mux SMC clock route register (CMXSMR)	R/W	8 bits	0x00	15.4.5/15-11
0x11B0D	Reserved	—	8 bits	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x11B0E	CPM mux UTOPIA address register (CMXUAR)	R/W	16 bits	0x0000	15.4.1/15-5
0x11B10–0x11B3F	Reserved	—	48 bytes	—	—
SI2 Registers					
0x11B40	SI2 TDMA2 mode register (SI2AMR)	R/W	16 bits	0x0000	14.5.2/14-15
0x11B42	SI2 TDMB2 mode register (SI2BMR)	R/W	16 bits	0x0000	
0x11B44	Reserved	R/W	24 bits	—	
0x11B48	SI2 global mode register (SI2GMR)	R/W	8 bits	0x00	14.5.1/14-15
0x11B49	Reserved	—	8 bits	—	—
0x11B4A	SI2 command register (SI2CMDR)	R/W	8 bits	0x00	14.5.4/14-22
0x11B4B	Reserved	—	8 bits	—	—
0x11B4C	SI2 status register (SI2STR)	R/W	8 bits	0x00	14.5.5/14-23
0x11B4D	Reserved	—	16 bits	—	—
0x11B4E	SI2 RAM shadow address register (SI2RSR)	R/W	16 bits	0x0000	14.5.3/14-21
0x11B50–0x11B5F	Reserved	—	16 bytes	—	—
USB					
0x11B60	USB mode register (USMOD)	R/W	8 bits	0x00	27.5.7.1/27-17
0x11B61	USB address register (USADR)	R/W	8 bits	0x00	27.5.7.2/27-18
0x11B62	USB command register (USCOM)	R/W	8 bits	0x00	27.5.7.4/27-20
0x11B64	USB end point 1 register (USEP1)	R/W	16 bits	0x0000	27.5.7.3/27-19
0x11B66	USB end point 2 register (USEP2)	R/W	16 bits	0x0000	27.5.7.3/27-19
0x11B68	USB end point 3 register (USEP3)	R/W	16 bits	0x0000	27.5.7.3/27-19
0x11B6A	USB end point 4 register (USEP4)	R/W	16 bits	0x0000	27.5.7.3/27-19
0x11B6C–0x11B6F	Reserved	—	32 bits	—	—
0x11B70	USB event register (USBER)	R/W	16 bits	0x0000	27.5.7.5/27-21
0x11B72	Reserved	—	16 bits	0x0000	—
0x11B74	USB mask register (USBMR)	R/W	16 bits	0x0000	27.5.7.6/27-22
0x11B77	USB status register (USBS)	R/W	8 bits	0x00	27.5.7.7/27-22
0x11B79–0x127FF	Reserved	—	3222 bytes	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
SI2 RAM					
0x12800–0x129FF	SI 2 transmit routing RAM (SI2TxRAM)	R/W	512 bytes	Undefined	14.4.3/14-9
0x12A00–0x12BFF	Reserved	—	512 bytes	—	—
0x12C00–0x12DFF	SI 2 receive routing RAM (SI2RxRAM)	R/W	512 bytes	Undefined	14.4.3/14-9
0x12E00–0x12FFF	Reserved	—	512 bytes	—	—
0x13000–0x13FFF	Reserved	—	4 Kbytes	—	—
CPM Instruction RAM					
0x20000–0x20FFF	CPM Instruction RAM (IRAM)	R/W	4 Kbytes	Undefined	
0x21000–0x2FFFF	Reserved	—	60 Kbytes	—	—
Reserved					
0x30000–0x3FFFF	Reserved	—	64 Kbytes	—	—
Security Engine (SEC)[1]					
0x40000–0x40FFF	Reserved[1]	—	4096 bytes	—	—
0x41000–0x41FFF	Controller[1]	R/W	4096 bytes	—	38.7/38-95
0x42000–0x42FFF	Crypto-channel 1[1]	R/W	4096 bytes	—	38.6/38-83
0x43000–0x43FFF	Crypto-channel 2[1]	R/W	4096 bytes	—	
0x44000–0x44FFF	Crypto-channel_3[1]	R/W	4096 bytes	—	
0x45000–0x45FFF	Crypto-channel_4[1]	R/W	4096 bytes	—	
0x46000–0x47FFF	Reserved[1]	—	8192 bytes	—	—
0x48000–0x48FFF	Arc Four execution unit[1]	R/W	4096 bytes	—	38.5.3/38-45
0x49000–0x49FFF	Reserved[1]	—	4096 bytes	—	—

Table 3-1. Internal Memory Map (continued)

Offset	Register	R/W	Size	Reset	Section/Page
0x4A000–0x 4AFFF	Data encryption standard execution unit[1]	—	4096 bytes	—	38.5.2/38-36
0x4B000–0x 4BFFF	Reserved[1]	—	4096 bytes	—	—
0x4C000–0x 4CFFF	Message digest execution unit[1]	R/W	4096 bytes	—	38.5.4/38-54
0x4D000–0x 4DFFF	Reserved[1]	—	4096 bytes	—	—
0x4E000–0x 4EFFF	Random number generator[1]	R/W	4096 bytes	—	38.5.5/38-64
0x4F000–0x 4FFFF	Reserved[1]	—	4096 bytes	—	—
0x50000–0x 50FFF	Public key execution unit[1]	R/W	4096 bytes	—	38.5.1/38-28
0x51000–0x 51FF	Reserved[1]	—	4096 bytes	—	—
0x52000–0x 52FFF	Advanced encryption standard execution unit[1]	R/W	4096 bytes	—	38.5.6/38-70

[1] MPC8272 and MPC8248 only. SEC modules are shown at the recommended offset IMMR + 0x4_0000. However, the offset for the SEC base address is selectable. The location of the SEC block must be mapped on 128-Kbyte resolution. Refer to Section 38.2.1, “SEC Address Base Register.”
For a complete list of SEC registers and their offsets and reset values, refer to Section 38.3, “Address Map,” in Chapter 38, “Security Engine (SEC).”

Part II
Configuration and Reset

Intended Audience

This part is intended for system designers and programmers who need to understand the operation of the MPC8272 at start up. It assumes an understanding of the PowerPC programming model described in the previous chapters and a high-level understanding of the MPC8272.

Contents

This part describes the start-up behavior of the MPC8272.

It contains the following chapters:

- Chapter 4, “System Interface Unit (SIU),” describes the system configuration and protection functions, which provide various monitors and timers, and the 60x bus configuration.
- Chapter 5, “Reset,” describes the behavior of the MPC8272 at reset and startup.

Suggested Reading

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.freescale.com. This documentation includes technical specifications, reference materials, and detailed application notes.

Conventions

This document uses the following notational conventions:

Bold — Bold entries in figures and tables showing registers and parameter RAM should be initialized by the user.

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.

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 or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.

x	In certain contexts, such as in a signal encoding or a bit field, indicates a don’t care.
n	Indicates an undefined numerical value

Acronyms and Abbreviations

Table II-1 contains acronyms and abbreviations that are used in this document. Note that the meanings for some acronyms (such as SDR1 and DSISR) are historical, and the words for which an acronym stands may not be intuitively obvious.

Table II-1. Acronyms and Abbreviated Terms

Term	Meaning
BIST	Built-in self test
DMA	Direct memory access
DRAM	Dynamic random access memory
EA	Effective address
GPR	General-purpose register
IEEE	Institute of Electrical and Electronics Engineers
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MSB	Most-significant byte
msb	Most-significant bit
MSR	Machine state register
PCI	Peripheral component interconnect
RTOS	Real-time operating system
Rx	Receive
SPR	Special-purpose register
SWT	Software watchdog timer
Tx	Transmit

Chapter 4
System Interface Unit (SIU)

The system interface unit (SIU) consists of several functions that control system start-up and initialization, as well as operation, protection, and the external system bus. Key features of the SIU include the following:

- System configuration and protection
- System reset monitoring and generation
- Clock synthesizer
- Power management
- 60x bus interface
- Flexible, high-performance memory controller
- Level 2 cache controller interface
- PCI interface
- IEEE 1149.1 test-access port (TAP)

Figure 4-1 is a block diagram of the SIU.

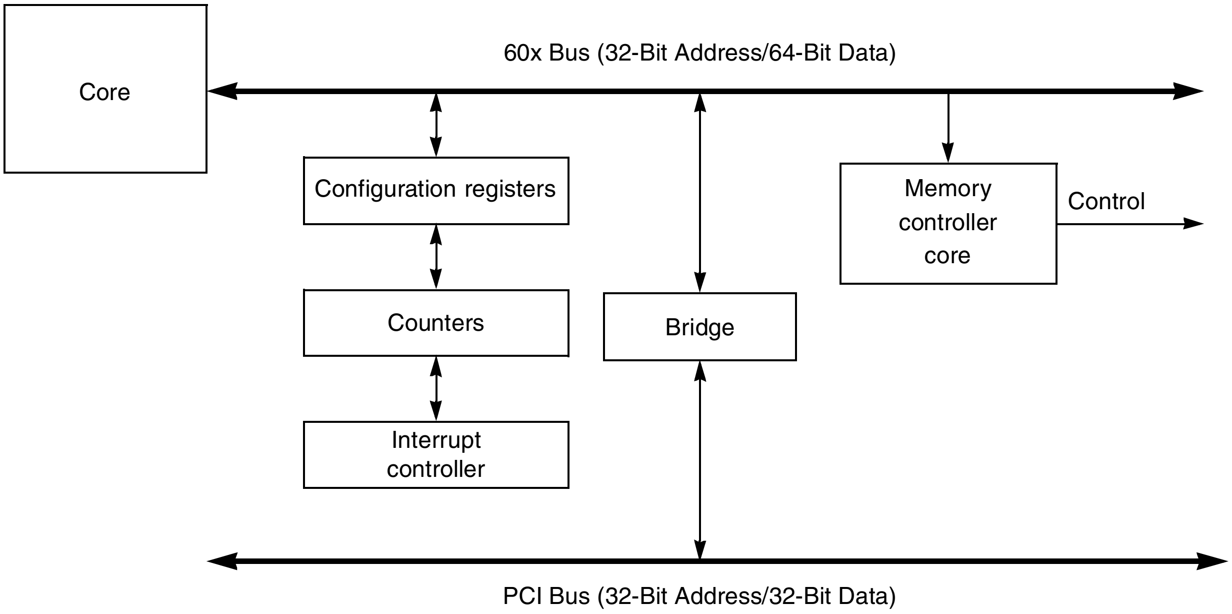


Figure 4-1. SIU Block Diagram

The system configuration and protection functions provide various monitors and timers, including the bus monitor, software watchdog timer, periodic interrupt timer, and time counter. The clock synthesizer generates the clock signals used by the SIU and other MPC8272 modules. The SIU clocking scheme supports stop and normal modes.

The 60x bus interface is a standard pipelined bus. The MPC8272 allows external bus masters to request and obtain system bus mastership. Chapter 8, “The 60x Bus,” describes bus operation, but this section explains 60x bus configuration.

The memory controller module, described in Chapter 11, “Memory Controller,” provides a seamless interface to many types of memory devices and peripherals. It supports up to eight memory banks, each with its own device and timing attributes. The PCI interface enables the use of standard peripherals.

The MPC8272’s implementation supports circuit board test strategies through a user- accessible test logic that is fully compliant with the IEEE 1149.1 test access port.

4.1 System Configuration and Protection

The MPC8272 incorporates many system functions that normally must be provided in external circuits. In addition, it is designed to provide maximum system safeguards against hardware and/or software faults. Table 4-1 describes the functions provided in the system configuration and protection submodule.

Table 4-1. System Configuration and Protection Functions

Function	Description
System configuration	The SIU allows the user to configure the system according to the particular requirements.
60x bus monitor	Monitors the transfer acknowledge ($\overline{\text{TA}}$) and address acknowledge ($\overline{\text{AACK}}$) response time for all bus accesses initiated by internal or external masters. $\overline{\text{TEA}}$, a core machine check or a system reset is asserted if the response limit is exceeded. This function can be disabled if needed.
Software watchdog timer	Asserts a reset or a core machine check, selected by the system protection control register (SYPCR) 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). After a system reset, this function is enabled, selects a maximum time-out period, and asserts a system reset if the time-out is reached. The software watchdog timer can be disabled or its time-out period may be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset. For more information, see Section 4.1.5, “Software Watchdog Timer.”
Periodic interrupt timer (PIT)	Generates periodic interrupts for use with a real-time operating system or the application software. The periodic interrupt timer (PIT) is clocked by the timersclk clock, providing a period from 122 µs to 8 seconds. The PIT function can be disabled if needed. See Section 4.1.4, “Periodic Interrupt Timer (PIT).”
Time counter	Provides a time-of-day information to the operating system/application software. It is composed of a 45-bit counter and an alarm register. A maskable interrupt is generated when the counter reaches the value programmed in the alarm register. The time counter (TMCNT) is clocked by the timersclk clock. See Section 4.1.3, “Time Counter (TMCNT).”

Figure 4-2 is a block diagram of the system configuration and protection logic.

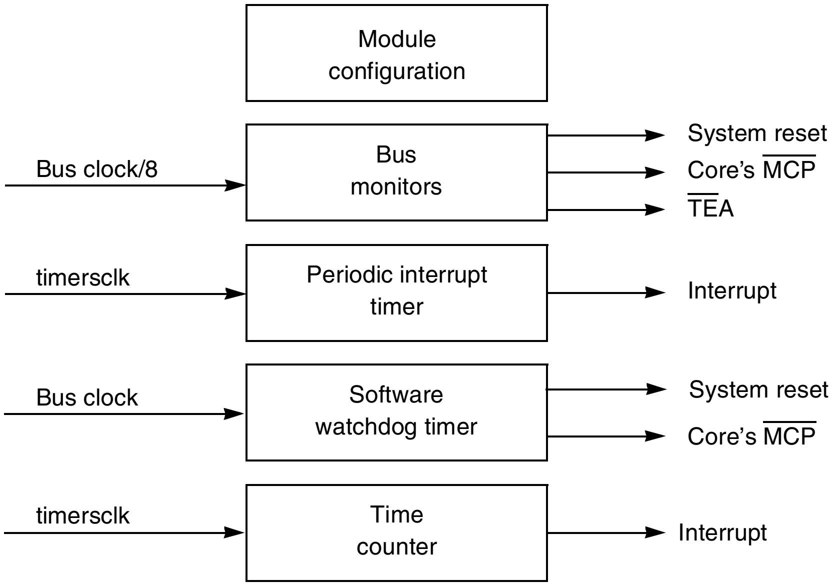


Figure 4-2. System Configuration and Protection Logic

Many aspects of system configuration are controlled by several SIU module configuration registers, described in Section 4.3.2, "System Configuration and Protection Registers."

4.1.1 Bus Monitor

The bus monitor ensures that each bus cycle is terminated within a reasonable period. The bus monitor does not count when the bus is idle. When a transaction starts ($\overline{\text{TS}}$ asserted), the bus monitor starts counting down from the time-out value. For standard bus transactions with an address tenure and a data tenure, the bus monitor counts until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the count down. This process continues until the whole data tenure is completed. Following the data tenure the bus monitor will idle in case there is no pending transaction; otherwise it will reload the time-out value and resume counting.

For address-only transactions, the bus monitor counts until $\overline{\text{AACK}}$ is asserted. If the monitor times out for a standard bus transaction, transfer error acknowledge ($\overline{\text{TEA}}$) is asserted. If the monitor times out for an address-only transaction, the bus monitor asserts $\overline{\text{AACK}}$ and a core machine check or reset interrupt is generated, depending on SYPCR[SWRI]. To allow variation in system peripheral response times, SYPCR[BMT] defines the time-out period, whose maximum value can be 2,040 system bus clocks. The timing mechanism is clocked by the system bus clock divided by eight.

4.1.2 Timers Clock

The two SIU timers (the time counter and the periodic interrupt timer) use the same clock source, timersclk, which can be derived from several sources, as described in Figure 4-3.

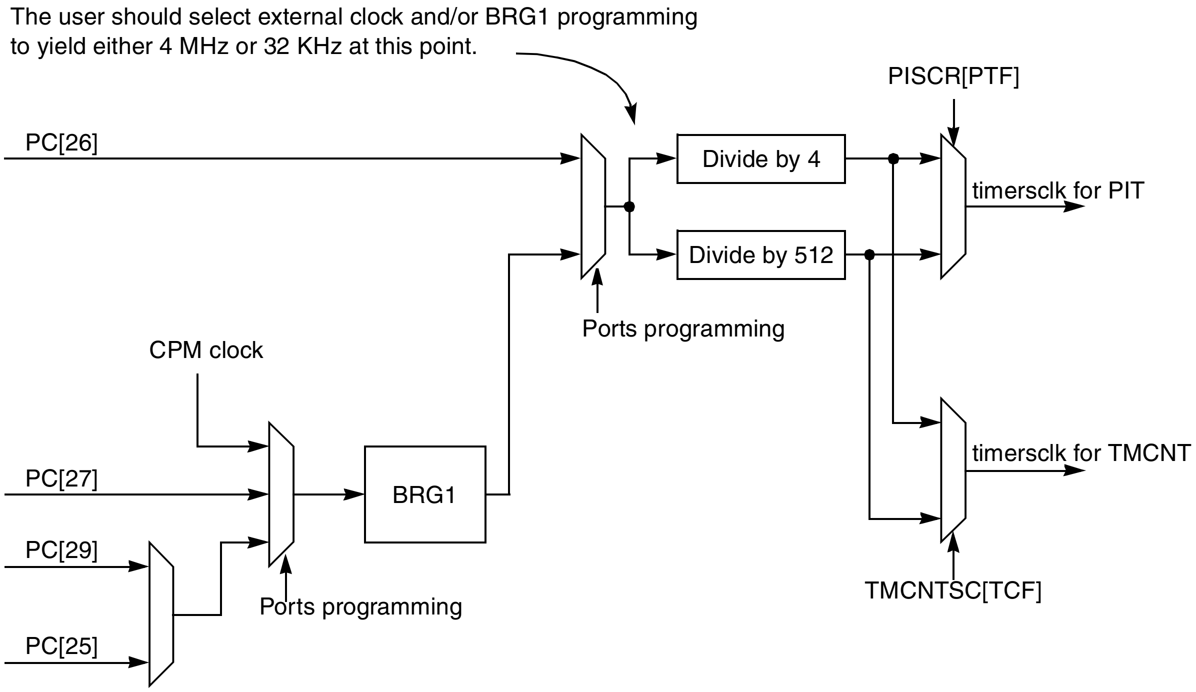


Figure 4-3. Timers Clock Generation

For details, see Section 37.2.4, “Port Pin Assignment Registers (PPARx).” For proper time counter operation, the user must ensure that the frequency of timersclk for TMCNT is 8,192 Hz by properly selecting the external clock and programming BRG1 and the prescalar control bits in the time counter status and control register (TMCNTSC[TCF]) and periodic interrupt status and control register (PISCR[PTF]).

4.1.3 Time Counter (TMCNT)

The time counter (TMCNT) is a 32-bit counter that is clocked by timersclk. It provides a time-of-day indication to the operating system and application software. The counter is reset to zero on $\overline{\text{PORESET}}$ reset or hard reset but is not affected by soft reset. It is initialized by the software; the user should set the timersclk frequency to 8,192 Hz, as explained in Section 4.1.2, “Timers Clock.”

TMCNT can be programmed to generate a maskable interrupt when the time value matches the value in its associated alarm register. It can also be programmed to generate an interrupt every second. The time counter control and status register (TMCNTSC) is used to enable or disable the various timer functions and report the interrupt source. Figure 4-4 shows a block diagram of TMCNT.

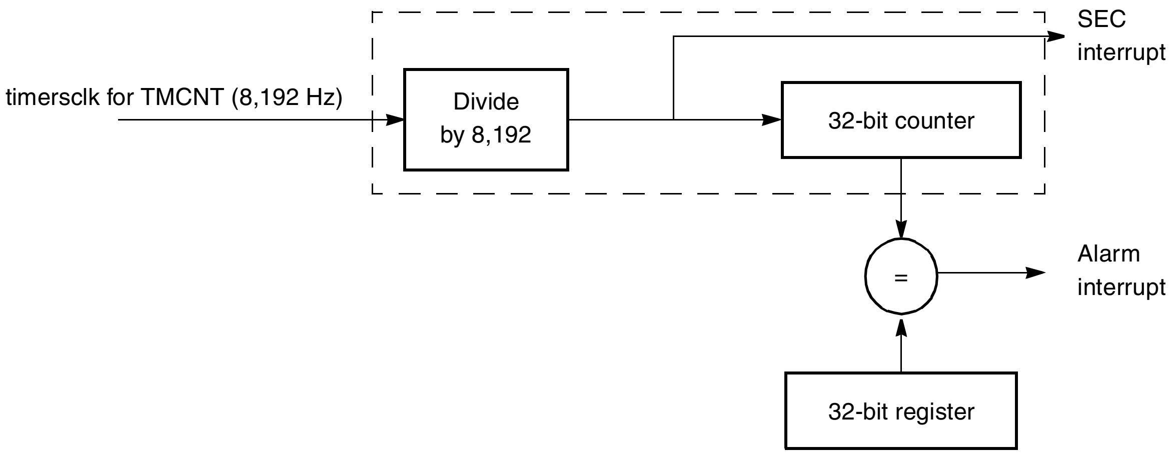


Figure 4-4. TMCNT Block Diagram

Section 4.3.2.11, “Time Counter Register (TMCNT),” describes the time counter register.

4.1.4 Periodic Interrupt Timer (PIT)

The periodic interrupt timer consists of a 16-bit counter clocked by timersclk. The 16-bit counter decrements to zero when loaded with a value from the periodic interrupt timer count register (PITC); after the timer reaches zero, PISCR[PS] is set and an interrupt is generated if PISCR[PIE] = 1. At the next input clock edge, the value in the PITC is loaded into the counter and the process repeats. When a new value is loaded into the PITC, the PIT is updated, the divider is reset, and the counter begins counting.

Setting PS creates a pending interrupt that remains pending until PS is cleared. If PS is set again before being cleared, the interrupt remains pending until PS is cleared. Any write to the PITC stops the current countdown and the count resumes with the new value in PITC. If PTE = 0, the PIT cannot count and retains the old count value. The PIT is not affected by reads. Figure 4-5 is a block diagram of the PIT.

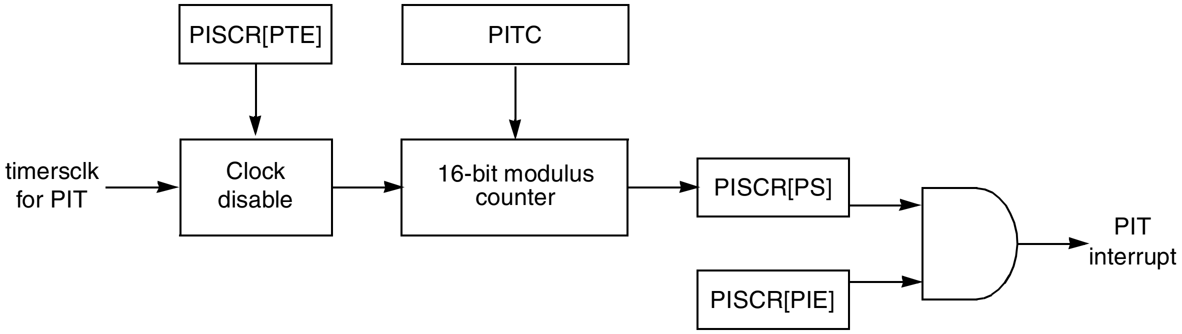


Figure 4-5. PIT Block Diagram

The time-out period is calculated as follows:

$$PIT_{period} = \frac{PITC + 1}{F_{timersclk}} = \frac{PITC + 1}{8192}$$

This gives a range from 122 µs (PITC = 0x0000) to 8 seconds (PITC = 0xFFFF).

4.1.5 Software Watchdog Timer

The SIU provides the software watchdog timer option to prevent system lock in case the software becomes trapped in loops with no controlled exit. Watchdog timer operations are configured in the SYPCR, which is described in Section 4.3.2.6, “System Protection Control Register (SYPCR).”

The software watchdog timer is enabled after reset to cause a hard reset if it times out. If the software watchdog timer is not needed, the user must clear SYPCR[SWE] 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 that is programmed in SYPCR[SWRI]. Once software writes SWRI, the state of SWE cannot be changed.

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

1. Write 0x556C to the software service register (SWSR).
2. Write 0xAA39 to SWSR.

The service sequence clears the watchdog timer and the timing process begins again. If a value other than 0x556C or 0xAA39 is written to the SWSR, 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 4-6 shows a state diagram for the watchdog timer.

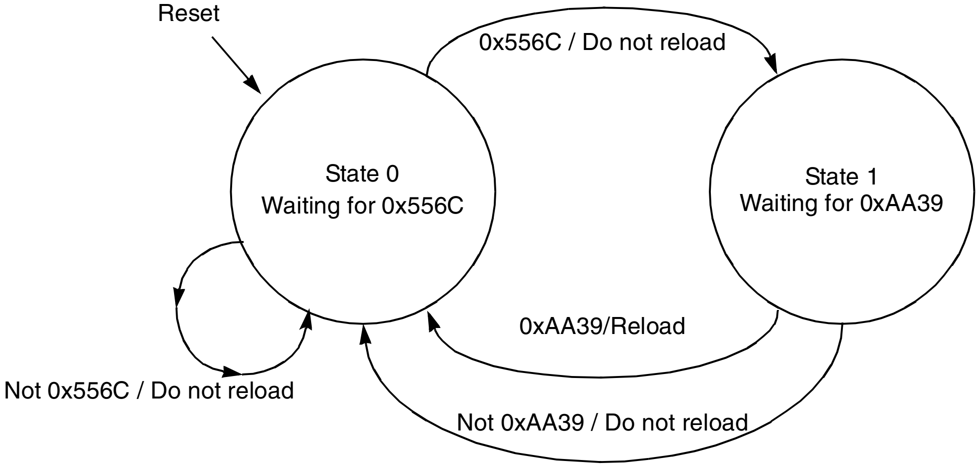


Figure 4-6. Software Watchdog Timer Service State Diagram

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 4-7 shows how to handle this need.

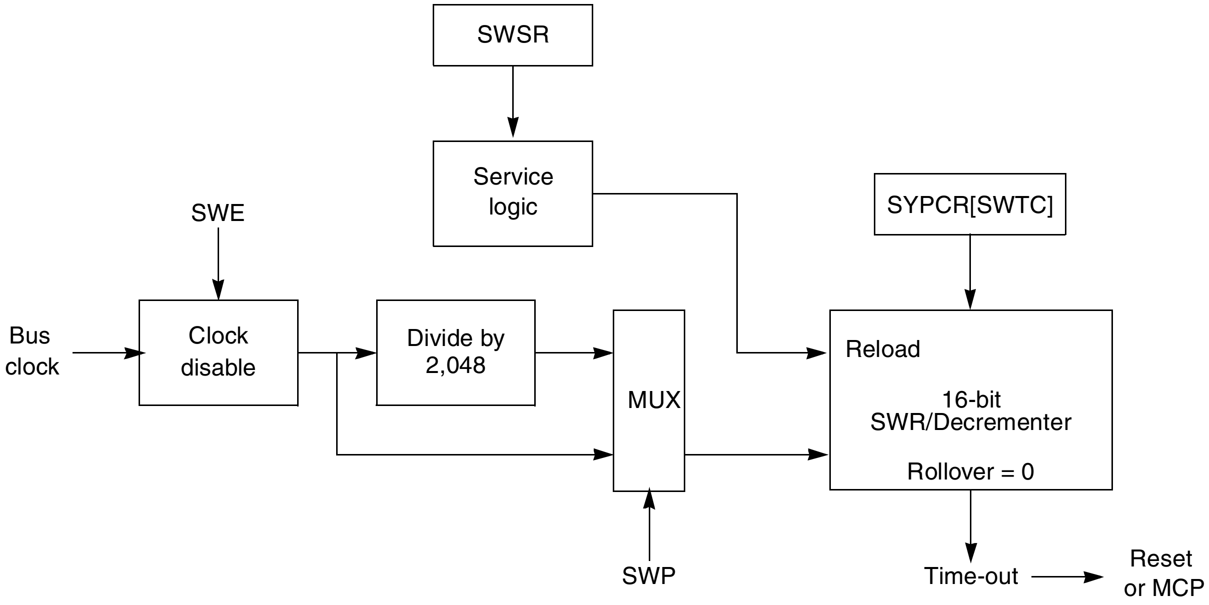


Figure 4-7. Software Watchdog Timer Block Diagram

In Figure 4-7, the range is determined by SYPCR[SWTC]. The value in SWTC is then loaded into a 16-bit decrementer clocked by the system clock. An additional divide-by-2,048 prescalar 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 $\overline{\text{MCP}}$ control logic. Upon reset, SWTC is set to the maximum value and is loaded again into the software watchdog register (SWR), 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 SWSR. If SYPCR[SWE] is loaded with 0, the modulus counter does not count.

4.2 Interrupt Controller

Key features of the interrupt controller include the following:

- Communications processor module (CPM) interrupt sources (FCCs, SCCs, timers, SMCs, I^{2}C, IDMA, SDMA, USB, and SPI)
- SIU interrupt sources (PIT, TMCNT, and PCI)
- 24 external sources (16 port C and 8 IRQ)
- Programmable priority between PIT, TMCNT, and PCI
- Programmable priority between SCCs and FCCs
- Two priority schemes for the SCCs: grouped, spread
- Programmable highest priority request
- Unique vector number for each interrupt source

4.2.1 Interrupt Configuration

Figure 4-8 shows the MPC8272 interrupt structure. The interrupt controller receives interrupts from internal sources, such as the PIT or TMCNT, from the CPM, the PCI bridge (with its own interrupt controller), and from external pins (port C parallel I/O pins and IRQ pins).

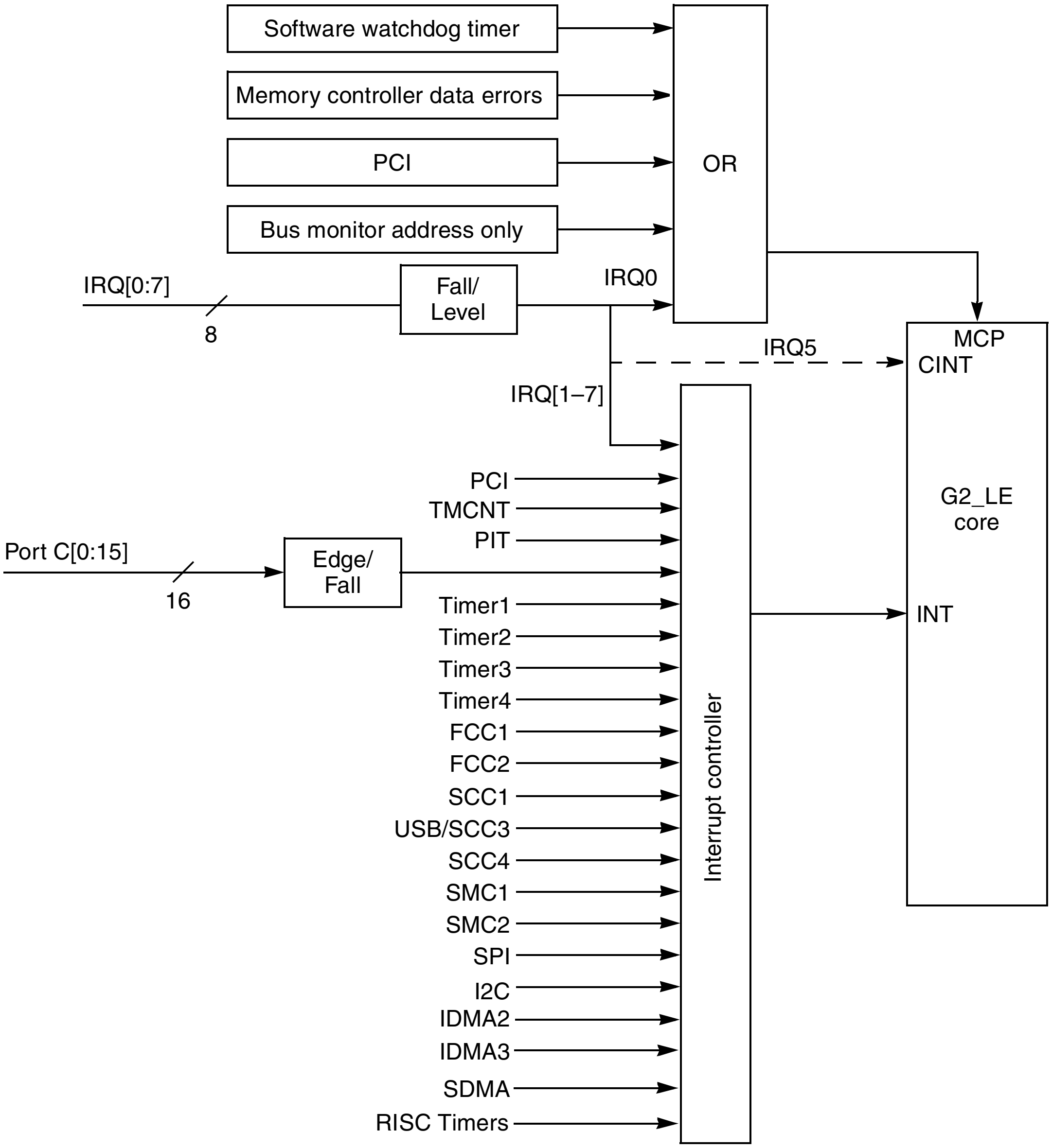


Figure 4-8. PowerQUICC II Interrupt Structure

The core takes a machine check interrupt when $\overline{\text{MCP}}$ is asserted (see Section 4.2.1.1, “Machine Check Interrupt (MCP)”), a critical interrupt when $\overline{\text{CINT}}$ is asserted (see Section 4.2.1.3, “Critical Interrupt (CINT)”), and an external interrupt when $\overline{\text{INT}}$ is asserted (see Section 4.2.1.2, “External Interrupt (INT)”).

4.2.1.1 Machine Check Interrupt (MCP)

There are several sources for a machine check interrupt:

- Software watchdog timer—See Section 4.1.5, “Software Watchdog Timer.”
- IRQ0 external interrupt
- PCI bridge
- Bus monitor time out for an address only transaction—See Section 4.1.1, “Bus Monitor.”

When the internal core is enabled, these sources cause a machine check interrupt assertion to the core. When the core is disabled the machine check interrupt assertion is reflected on IRQ0/NMI_OUT for the use of an external processor. A machine check interrupt is taken only if the machine check enable bits in the machine state register (MSR[ME]) and in the hardware implementation register 0 (HID0[EMCP]) are both set.

4.2.1.2 External Interrupt (INT)

The interrupt controller allows masking of each external interrupt source. Multiple events within a CPM sub-block event are also maskable.

All external interrupt sources are prioritized and bits are set in the interrupt pending register (SIPNR). On the MPC8272, the prioritization of the interrupt sources is flexible in the following two aspects:

- The relative priority of the FCCs and SCCs can be modified.
- One interrupt source can be assigned the highest priority.

When an unmasked interrupt source is pending in the SIPNR, the interrupt controller sends an interrupt request to the core. An external interrupt is taken only if the external interrupt enable bit in the machine state register (MSR[EE]) is set. When an exception is taken, the external interrupt enable bit in the machine state register (MSR[EE]) is cleared to disable further interrupt requests until software can handle them.

The SIU interrupt vector register (SIVEC) is updated with a 6-bit vector corresponding to the sub-block with the highest current priority.

The interrupt controller allows masking of each interrupt source. Multiple events within a CPM sub-block event are also maskable.

If the internal core is disabled, INT is reflected on CPU_BR/INT_OUT for the use of an external processor.

4.2.1.3 Critical Interrupt (CINT)

The critical interrupt can be asserted by the following pins:

- IRQ5/CINT/TBEN/EXT_DBG3
- BADDR31/IRQ5/CINT

A critical interrupt is taken only if the critical interrupt exception enable bit in the machine state register (MSR[CE]) is set.

Note that IRQ5 and CINT cannot be used at the same time; when IRQ5 is used CINT should be disabled by clearing the critical interrupt exception enable bit in the machine state register (MSR[CE]); when CINT is used $\overline{\text{IRQ5}}$ should be disabled by clearing the IRQ5 mask bit in the SIU interrupt mask register (SIMR_H[IRQ5]).

4.2.1.4 External Interrupt Sources in Single MPC8272 Bus Mode

The following external interrupt sources become available when using single MPC8272 bus mode:

- $\overline{\text{IRQ1}}$ from the pin BADDR[27]/$\overline{\text{IRQ1}}$
- $\overline{\text{IRQ2}}$ from the pin BADDR[28]/$\overline{\text{IRQ2}}$
- $\overline{\text{IRQ4}}$ from the pin ALE/$\overline{\text{IRQ4}}$
- $\overline{\text{IRQ6}}$ from the pin $\overline{\text{BG\_B}}$/$\overline{\text{IRQ6}}$
- $\overline{\text{IRQ7}}$ from the pin $\overline{\text{DBG\_B}}$/$\overline{\text{IRQ7}}$

4.2.2 Interrupt Source Priorities

The interrupt controller has 44 interrupt sources that assert one interrupt request to the core. Table 4-2 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 as shown. A single interrupt priority number is associated with each table entry.

Note that the group and spread options, shown with YCC entries in Table 4-2, are described in Section 4.2.2.1, “SCC and FCC Relative Priority.”

Table 4-2. Interrupt Source Priority Levels

Priority Level	Interrupt Source Description	Multiple Events
1	Highest	—
2	XSIU1	No (TMCNT,PIT,PCI = Yes)
3	XSIU2 (Grouped)	No (TMCNT,PIT,PCI = Yes)
4	XSIU3 (Grouped)	No (TMCNT,PIT,PCI = Yes)
5	XSIU4 (Grouped)	No (TMCNT,PIT,PCI= Yes)
6	XCC1	Yes
7	XCC2	Yes
8	XCC3	Yes
9	XCC4	Yes
10	XSIU2 (Spread)	No (TMCNT,PIT,PCI = Yes)
11	XCC5	Yes
12	XCC6	Yes
13	XCC7	Yes
14	XCC8	Yes
15	XSIU5 (Grouped)	No (TMCNT,PIT,PCI = Yes)
16	XSIU6 (Grouped)	No (TMCNT,PIT,PCI = Yes)

Table 4-2. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
17	XSIU7 (Grouped)	No (TMCNT,PIT,PCI = Yes)
18	XSIU8 (Grouped)	No (TMCNT,PIT,PCI = Yes)
19	XSIU3 (Spread)	No (TMCNT,PIT,PCI = Yes)
20	YCC1 (Grouped)	Yes
21	YCC2 (Grouped)	Yes
22	YCC3 (Grouped)	Yes
23	YCC4 (Grouped)	Yes
24	YCC5 (Grouped)	Yes
25	YCC6 (Grouped)	Yes
26	YCC7 (Grouped)	Yes
27	YCC8 (Grouped)	Yes
28	XSIU4 (Spread)	No (TMCNT,PIT,PCI = Yes)
29	Parallel I/O–PC29	Yes
30	Timer 1	Yes
31	Parallel I/O–PC23	Yes
32	YCC1 (Spread)	Yes
33	Parallel I/O–PC15	Yes
34	SDMA Bus Error	Yes
35	USB	Yes
36	Reserved	—
37	Parallel I/O–PC14	No
38	Parallel I/O–PC13	No
39	IDMA2	Yes
40	Timer 2	Yes
41	Parallel I/O–PC12	No
42	XSIU5 (GSIU = 1)	No (TMCNT,PIT,PCI = Yes)
43	YCC3 (Spread)	Yes
44	RISC Timer Table	Yes
45	I2C	Yes
46	YCC4 (Spread)	Yes
47	Parallel I/O–PC11	No
48	Parallel I/O–PC10	No
49	IRQ6	No

Table 4-2. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
50	IDMA3	Yes
51	IRQ7	No
52	Timer 3	Yes
53	XSIU6 (GSIU = 1)	No (TMCNT,PIT,PCI = Yes)
54	YCC5 (Spread)	Yes
55	Parallel I/O–PC9	No
56	Parallel I/O–PC8	No
57	Parallel I/O–PC7	No
58	Timer 4	Yes
59	YCC6 (Spread)	Yes
60	Parallel I/O–PC6	No
61	XSIU7 (GSIU = 1)	No (TMCNT,PIT,PCI = Yes)
62	Reserved	--
63	SPI	Yes
64	Parallel I/O–PC5	No
65	Parallel I/O–PC4	No
66	SMC1	Yes
67	YCC7 (Spread)	Yes
68	SMC2	Yes
69	Parallel I/O–PC1	No
70	Parallel I/O–PC0	No
71	XSIU8 (GSIU = 1)	No (TMCNT,PIT,PCI = Yes)
72	YCC8 (Spread)	Yes
73	Reserved	—

Notice the lack of SDMA interrupt sources, which are reported through each individual FCC, SCC, SMC, SPI, or I^{2}C channel. The only true SDMA interrupt source is the SDMA channel bus error entry that is reported when a bus error occurs during an SDMA access. There are two ways to add flexibility to the table of CPM interrupt priorities—the FCC and SCC relative priority option, described in Section 4.2.2.1, “SCC and FCC Relative Priority,” and the highest priority option, described in Section 4.2.2.3, “Highest Priority Interrupt.”

4.2.2.1 SCC and FCC Relative Priority

The relative priority between the three SCCs and two FCCs is programmable and can be changed dynamically. In Table 4-2 there is no entry for SCC1. SCC3–SCC4, FCC1–FCC2, but rather there are entries for XCC1–XCC8 and YCC1–YCC8. Each SCC can be mapped to any YCC location and each FCC can be mapped to any XCC location. The SCC and FCC priorities are programmed in the CPM interrupt priority registers (SCPRR_H and SCPRR_L) and can be changed dynamically to implement a rotating priority.

In addition, there are two options for the grouping of the locations of the YCC entries:

- Group—In this scheme, all SCCs are grouped together at the top of the priority table, ahead of most other CPM interrupt sources. This scheme is ideal for applications where all SCCs and FCCs function at a very high data rate and interrupt latency is very important.
- Spread—In this scheme, priorities are spread over the table so other sources can have lower interrupt latencies. This scheme is also programmed in the SICR but cannot be changed dynamically.

4.2.2.2 PIT, TMCNT, PCI, and IRQ Relative Priority

The MPC8272 has seven general-purpose interrupt requests (IRQs), five of which, along with the PIT, the PCI interrupt controller, and TMCNT, can be mapped to any XSIU location. $\overline{\text{IRQ6}}$ and $\overline{\text{IRQ7}}$ have fixed priority.

4.2.2.3 Highest Priority Interrupt

In addition to the FCC/SCC relative priority option, SICR[HP] can be used to specify one interrupt source as having highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in the table.

If the highest priority feature is not used, select the interrupt request in XSIU1 to be the highest priority interrupt; the standard interrupt priority order is used. SICR[HP] can be updated dynamically to allow the user to change a normally low priority source into a high priority-source for a certain period.

4.2.3 Masking Interrupt Sources

By programming the SIU mask registers, SIMR_H and SIMR_L, the user can mask interrupt requests to the core. Each SIMR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding SIMR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIPNR bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

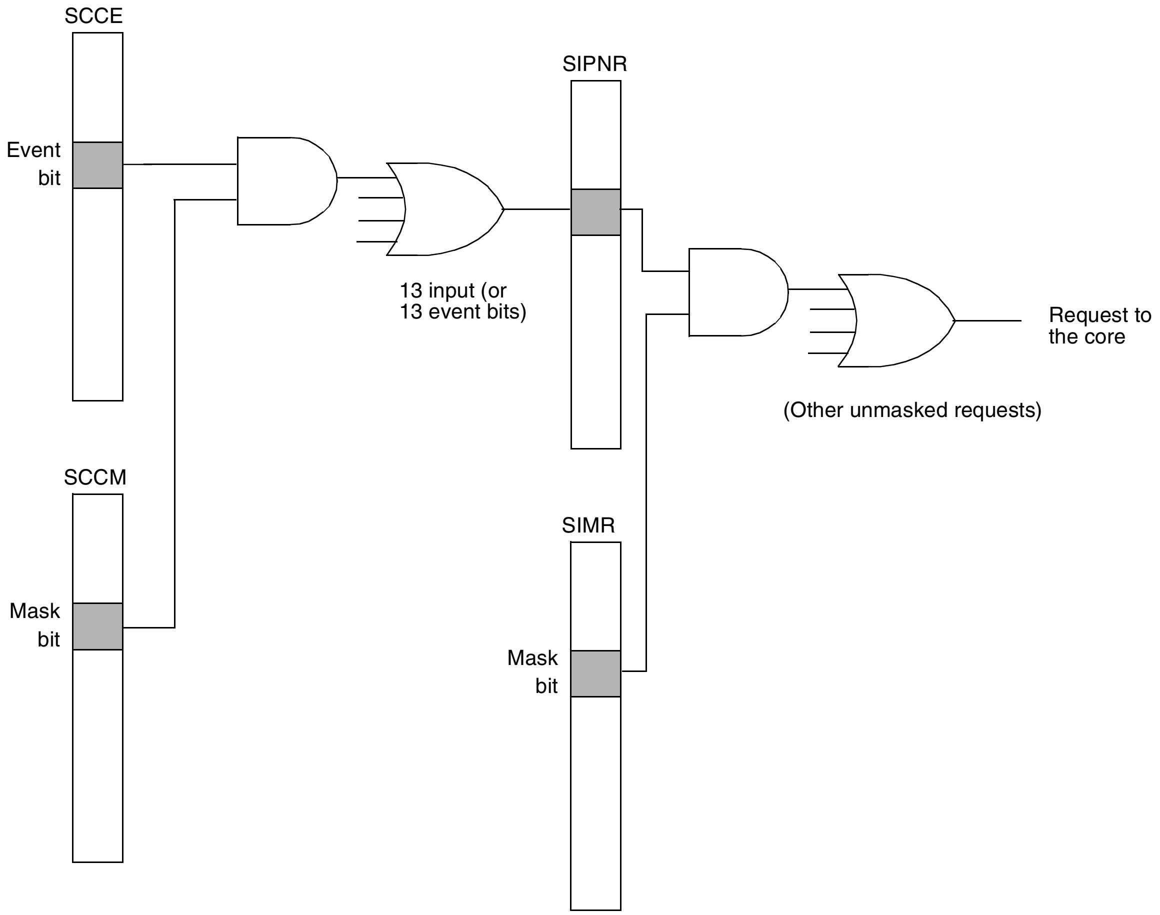


Figure 4-9. Interrupt Request Masking

4.2.4 Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by reading SIVEC. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt. Table 4-3 lists encodings for the six low-order bits of the interrupt vector.

Table 4-3. Encoding the Interrupt Vector

Interrupt Number	Interrupt Source Description	Interrupt Vector
0	Error (No interrupt)	0b00_0000
1	I^{2}C	0b00_0001
2	SPI	0b00_0010
3	RISC timers	0b00_0011
4	SMC1	0b00_0100
5	SMC2	0b00_0101

Table 4-3. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
6	Reserved	0b00_0110
7	IDMA2	0b00_0111
8	IDMA3	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	TMCNT	0b01_0000
17	PIT	0b01_0001
18	PCI	0b01_0010
19	IRQ1	0b01_0011
20	IRQ2	0b01_0100
21	IRQ3	0b01_0101
22	IRQ4	0b01_0110
23	IRQ5	0b01_0111
24	IRQ6	0b01_1000
25	IRQ7	0b01_1001
26–31	Reserved	0b01_1010–01_1111
32	FCC1	0b10_0000
33	FCC2	0b10_0001
34	Reserved	0b10_0010
35	Reserved	0b10_0011
36	Reserved	0b10_0100
37	Reserved	0b10_0101
38	Reserved	0b10_0110
39	Reserved	0b10_0111
40	SCC1	0b10_1000
41	Reserved	0b10_1001
42	SCC3	0b10_1010
43	SCC4	0b10_1011

Table 4-3. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
44–46	Reserved	0b10_1100–0b10_1110
47	SEC	0b10_1111
48	PC29	0b11_0000
49	PC23	0b11_0001
50	PC15	0b11_0010
51	PC14	0b11_0011
52	PC13	0b11_0100
53	PC12	0b11_0101
54	PC11	0b11_0110
55	PC10	0b11_0111
56	PC9	0b11_1000
57	PC8	0b11_1001
58	PC7	0b11_1010
59	PC6	0b11_1011
60	PC5	0b11_1100
61	PC4	0b11_1101
62	PC1	0b11_1110
63	PC0	0b11_1111

Note that the interrupt vector table differs from the interrupt priority table in only two ways:

- FCC and SCC vectors are fixed; they are not affected by the SCC group mode, spread mode, or the relative priority order of the FCCs and SCCs.
- An error vector exists as the last entry in Table 4-3. The error vector is issued when no interrupt is requesting service.

4.2.4.1 Port C External Interrupts

There are 16 external interrupts issued from the parallel I/O port C pins, PC[0:1, 4:15, 23, 29]. When ones of these pins is configured as an input, a change according to the SIU external interrupt control register (SIEXR) causes an interrupt request signal to be sent to the interrupt controller. PC[0:1, 4:15, 23, 29] lines can be programmed to assert an interrupt request upon any change. Each port C line asserts a unique interrupt request to the interrupt pending register and has a different internal interrupt priority level within the interrupt controller.

Requests can be masked independently in the interrupt mask register (SIMR). Notice that the global SIMR is cleared on system reset so that pins left floating do not cause false interrupts.

4.3 Programming Model

The SIU registers are grouped into the following three categories:

- Interrupt controller registers. These registers control the configuration, prioritization, and masking of interrupts. They also include registers for determining the interrupt sources. These registers are described in Section 4.3.1, "Interrupt Controller Registers."
- System configuration and protection registers. These include registers for configuring the SIU, defining the base address for the internal memory map, configuring the watchdog timer, and specifying bus characteristics, and the general functionality of the 60x bus, such as arbitration, error status, and control. These registers are described in Section 4.3.2, "System Configuration and Protection Registers."
- Periodic interrupt registers. These include registers for configuring and providing status for periodic interrupts. See Section 4.3.3, "Periodic Interrupt Registers."

4.3.1 Interrupt Controller Registers

There are seven interrupt controller registers, described in the following sections:

- Section 4.3.1.1, "SIU Interrupt Configuration Register (SICR)"
- Section 4.3.1.2, "SIU Interrupt Priority Register (SIPRR)"
- Section 4.3.1.3, "CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L)"
- Section 4.3.1.4, "SIU Interrupt Pending Registers (SIPNR_H and SIPNR_L)"
- Section 4.3.1.5, "SIU Interrupt Mask Registers (SIMR_H and SIMR_L)"
- Section 4.3.1.6, "SIU Interrupt Vector Register (SIVEC)"
- Section 4.3.1.7, "SIU External Interrupt Control Register (SIEXR)"

4.3.1.1 SIU Interrupt Configuration Register (SICR)

The SIU interrupt configuration register (SICR), shown in Figure 4-10, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table displayed in Table 4-2.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>7</th><th>8</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2">—</td><td colspan="2">HP</td><td colspan="2">—</td><td>GSIU</td><td>SPS</td></tr>
<tr><td>Reset</td><td colspan="8">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="8">R/W</td></tr>
<tr><td>Addr</td><td colspan="8">0x10C00</td></tr>
</table>

Figure 4-10. SIU Interrupt Configuration Register (SICR)

The SICR register bits are described in Table 4-4.

Table 4-4. SICR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–7	HP	Highest priority. Specifies the 6-bit interrupt number of the single interrupt controller interrupt source that is advanced to the highest priority in the table. HP can be modified dynamically. To retain the original priority, program HP to the interrupt number assigned to XSIU1.
8–13	—	Reserved, should be cleared.
14	GSIU	Group SIU. Selects the relative XSIU priority scheme. It cannot be changed dynamically. 0 Grouped. The XSIUs are grouped by priority at the top of the table. 1 Spread. The XSIUs are spread by priority in the table.
15	SPS	Spread priority scheme. Selects the relative YCC priority scheme. It cannot be changed dynamically. 0 Grouped. The YCCs are grouped by priority at the top of the table. 1 Spread. The YCCs are spread by priority in the table.

4.3.1.2 SIU Interrupt Priority Register (SIPRR)

The SIU interrupt priority register (SIPRR), shown in Figure 4-11, defines the priority between IRQ1:IRQ5, PIT, PCI, and TMCNT.

	0 2	3 5	6 8	9 11	12 15
Field	XS1P	XS2P	XS3P	XS4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Addr	0x10C10				

	16 18	19 21	22 24	25 27	28 31
Field	XS5P	XS6P	XS7P	XS8P	—
Reset	100	101	110	111	0000
R/W	R/W				
Addr	0x10C12				

Figure 4-11. SIU Interrupt Priority Register (SIPRR)

The SIPRR register bits are described in Table 4-5.

Table 4-5. SIPRR Field Descriptions

Bits	Name	Description
0–2	XS1P–XSIU1	Priority order. Defines which PIT/TMCNT/PCI/IRQs asserts its request in the XSIU1 priority position. The user should not program the same PIT/TMCNT/PCI/IRQs to more than one priority position (1–8). These bits can be changed dynamically. 000 TMCNT asserts its request in the XSIU1 position. 001 PIT asserts its request in the XSIU1 position. 010 PCI asserts its request in the XSIU1 position. 011 $\overline{\text{IRQ1}}$ asserts its request in the XSIU1 position. 100 $\overline{\text{IRQ2}}$ asserts its request in the XSIU1 position. 101 $\overline{\text{IRQ3}}$ asserts its request in the XSIU1 position. 110 $\overline{\text{IRQ4}}$ asserts its request in the XSIU1 position. 111 $\overline{\text{IRQ5}}$ asserts its request in the XSIU1 position.
3–11, 16–27	XS2P–XS8P	Same as XS1P, but for XSIU2–XSIU8
12–15, 28–31	—	Reserved, should be cleared.

4.3.1.3 CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L)

The CPM high interrupt priority register (SCPRR_H), shown in Figure 4-12, define priorities between the FCCs.

	0 2	3 5	6 8	9 11	12	13 15
Field	XC1P	XC2P	XC3P	XC4P	—	—
Reset	000	001	010	011	0	000
R/W	R/W					
Addr	0x10C14					

	16 18	19 21	22 24	25 27	28	29 31
Field	XC5P	XC6P	XC7P	XC8P	—	—
Reset	100	101	110	111	0	000
R/W	R/W					
Addr	0x10C16					

Figure 4-12. CPM High Interrupt Priority Register (SCPRR_H)

Table 4-6 describes the SCPRR_H fields.

Table 4-6. SCPRR_H Field Descriptions

Bits	Name	Description
0–2	XC1P–XCC1	Priority order. Defines which FCC asserts its request in the XCC1 priority position. The user should not program the same FCC to more than one priority position (1–8). These bits can be changed dynamically. 000 FCC1 asserts its request in the XCC1 position. 001 FCC2 asserts its request in the XCC1 position. 010 XCC1 position not active 011 XCC1 position not active 100 XCC1 position not active 101 XCC1 position not active 110 XCC1 position not active 111 XCC1 position not active
3–11	XC2P–XC4P	Same as XC1P, but for XCC2:XCC4
12–15	—	Reserved, should be cleared.
16–27	XC5P–XC8P	Same as XC1P, but for XCC5:XCC8
28–31	—	Reserved, should be cleared.

The CPM low interrupt priority register (SCPRR_L), shown in Figure 4-13, defines prioritization of SCCs.

	0 2	3 5	6 8	9 11	12 15
Field	YC1P	YC2P	YC3P	YC4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Addr	0x10C18				

	16 18	19 21	22 24	25 27	28 31
Field	YC5P	YC6P	YC7P	YC8P	—
Reset	100	101	110	111	0000
R/W	R/W				
Addr	0x10C20				

Figure 4-13. CPM Low Interrupt Priority Register (SCPRR_L)

Table 4-7 describes the SCPRR_L fields.

Table 4-7. SCPRR_L Field Descriptions

Bits	Name	Description
0–2	YC1P–YCC1	Priority order. Defines which SCC asserts its request in the YCC1 priority position. Do not program the same SCC to multiple priority positions. This field can be changed dynamically. 000 SCC1 asserts its request in the YCC1 position. 001 YCCI position is not active. 010 SCC3 asserts its request in the YCC1 position. 011 SCC4 asserts its request in the YCC1 position. 100 YCCI position is not active. 101 YCCI position is not active. 111 YCCI position is not active. 111 SEC asserts its requests in the YCC1 position.
3–11	YC2P–YC8P	Same as YC1P, but for YCC2–YCC8
12–15	—	Reserved, should be cleared.
16–27	YC5P–YC8P	Same as YC1P, but for YCC5–YCC8
28–31	—	Reserved, should be cleared.

4.3.1.4 SIU Interrupt Pending Registers (SIPNR_H and SIPNR_L)

Each bit in the interrupt pending registers (SIPNR_H and SIPNR_L), shown in Figure 4-14 and Figure 4-15, corresponds to an interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC23	PC29
Reset	Undefined (The user should write 1s to clear these bits before using.)															
R/W	R/W															
Addr	0x10C08															

	16	17	18	19	20	21	22	23	24 28	29	30	31
Field	—	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5/CINT	IRQ6	IRQ7	—	TMCNT	PIT	PCI
Reset	Undefined (The user should write 1s to clear these bits before using.)									0[1]	0[1]	0[1]
R/W	R/W											
Addr	0x10C10											

[1] These fields are zero after reset because their corresponding mask register bits are cleared (disabled).

Figure 4-14. SIU High Interrupt Pending Register (SIPNR_H)

Figure 4-15 shows the SIPNR_L fields.

	0	1	2	7	8	9	10	11	12	14	15
Field	FCC1	FCC2	—		SCC1	—	SCC3	SCC4	—		SEC
Reset	0000_0000_0000_0000[1]										
R/W	R/W										
Addr	0x10C0C										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	I2C	SPI	RTT	SMC1	SMC2	—	IDMA2	IDMA3	—	SDMA	USB	TIMER1	TIMER2	TIMER3	TIMER4	—
Reset	0000_0000_0000_000[1]															0
R/W	R/W															
Addr	0x10C0E															

[1] These fields are zero after reset because their corresponding mask register bits (SCCM) are cleared (disabled).

Figure 4-15. SIU Low Interrupt Pending Register (SIPNR_L)

When a pending interrupt is handled, the user clears the corresponding SIPNR bit. However, if an event register exists, the unmasked event register bits should be cleared instead, causing the SIPNR bit to be cleared.

SIPNR bits are cleared by writing ones to them. Because the user can only clear bits in this register, writing zeros to this register has no effect.

Note that the SCC/FCC SIPNR bit positions do not change according to their relative priority.

4.3.1.5 SIU Interrupt Mask Registers (SIMR_H and SIMR_L)

Each bit in the SIU interrupt mask register (SIMR) corresponds to an interrupt source. The user masks an interrupt by clearing and enables an interrupt by setting the corresponding SIMR bit. When a masked interrupt occurs, the corresponding SIPNR bit is set, regardless of the SIMR bit, although no interrupt request is passed to the core.

If an interrupt source requests interrupt service when the user clears its SIMR bit, the request stops. If the user sets the SIMR bit later, a previously pending interrupt request is processed by the core, according to its assigned priority. The SIMR can be read by the user at any time.

Figure 4-16 shows the SIMR_H register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC23	PC29
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10C1C															

	16	17	18	19	20	21	22	23	24	28	29	30	31
Field	—	IRQ1	IRQ2	IRQ3	IRQ4	IRQ5/CINT	IRQ6	IRQ7	—		TMCNT	PIT	PCI
Reset	0000_0000_0000_0000												
R/W	R/W												
Addr	0x10C1E												

Figure 4-16. SIU High Interrupt Mast Register (SIMR_H)

Figure 4-17 shows the SIMR_L register.

	0	1	2	7	8	9	10	11	12	14	15
Field	FCC1	FCC2	—		SCC1	—	SCC3	SCC4	—		SEC
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x10C20										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	I2C	SPI	RTT	SMC1	SMC2	—	IDMA2	IDMA3	—	SDMA	USB	TIMER1	TIMER2	TIMER3	TIMER4	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10C22															

Figure 4-17. SIU Low Interrupt Mast Register (SIMR_L)

Note the following:

- SCC/FCC SIMR bit positions are not affected by their relative priority.
- The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register.
- If an SIMR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Thus, the user should always include an error vector routine, even if it contains only an rfi instruction. The error vector cannot be masked.

4.3.1.6 SIU Interrupt Vector Register (SIVEC)

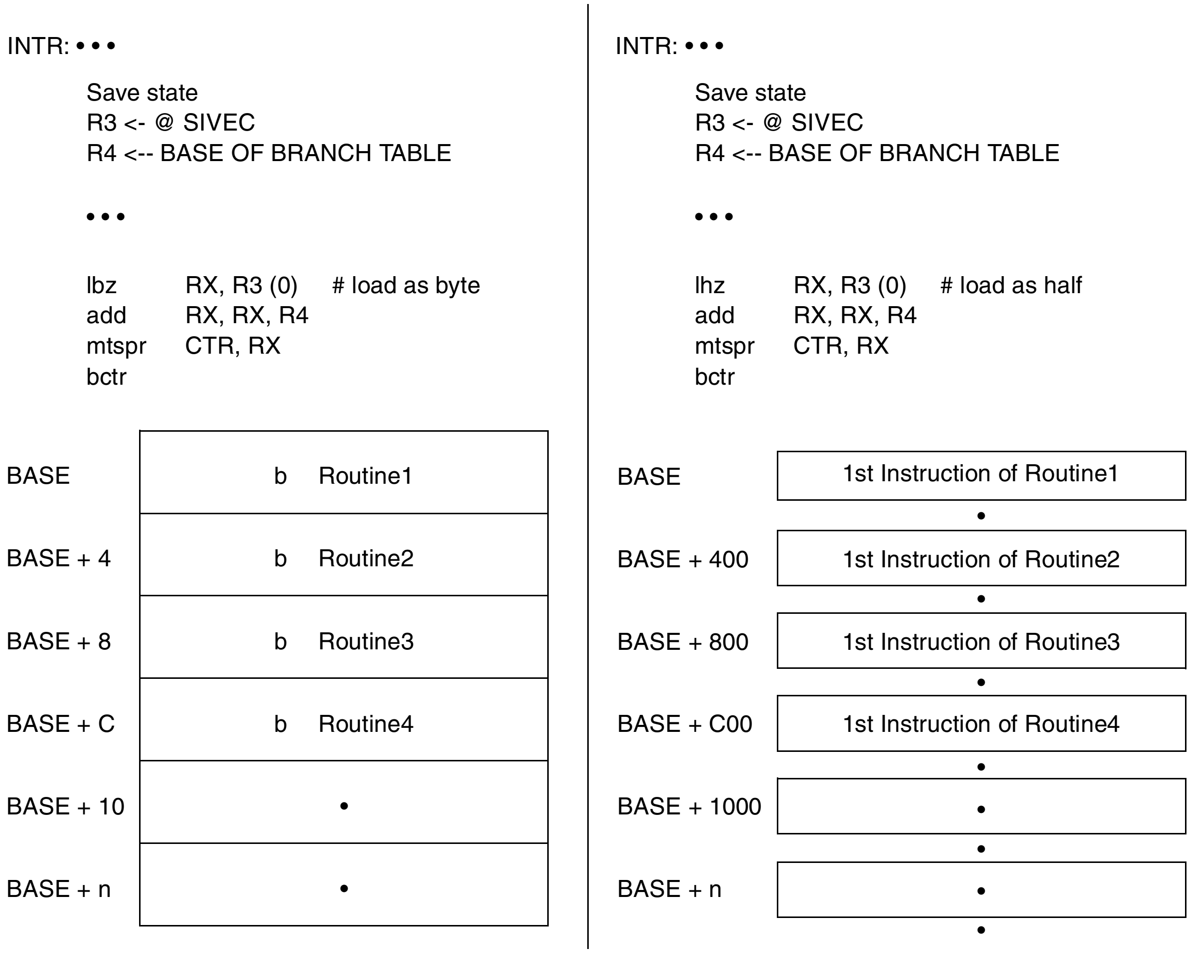
The SIU interrupt vector register (SIVEC), shown in Figure 4-18, contains an 8-bit code representing the unmasked interrupt source of the highest priority level.

	0	5	6	7	8	9	10	11	12	13	14	15
Field	Interrupt Code		0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000											
R/W	R											
Addr	0x10C04											

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000															
R/W	R															
Addr	0x10C06															

Figure 4-18. SIU Interrupt Vector Register (SIVEC)

The SIVEC 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 zeros, allowing indexing into the table, as shown in Figure 4-19.



```
INTR: • • •
          Save state
          R3 <- @ SIVEC
          R4 <-- BASE OF BRANCH TABLE

          • • •

          lbz      RX, R3 (0)    # load as byte
          add      RX, RX, R4
          mtspr    CTR, RX
          bctr
```

```
INTR: • • •
          Save state
          R3 <- @ SIVEC
          R4 <-- BASE OF BRANCH TABLE

          • • •

          lhz      RX, R3 (0)    # load as half
          add      RX, RX, R4
          mtspr    CTR, RX
          bctr
```

Figure 4-19. Interrupt Table Handling Example

NOTE

The MPC8272 differs from previous MPC8xx implementations in that when an interrupt request occurs, SIVEC can be read. If there are multiple interrupt sources, SIVEC latches the highest priority interrupt. Note that the value of SIVEC cannot change while it is being read.

4.3.1.7 SIU External Interrupt Control Register (SIEXR)

Each defined bit in the SIU external interrupt control register (SIEXR), shown in Figure 4-20, determines whether the corresponding port C line asserts an interrupt request upon either a high-to-low change or any change on the pin. External interrupts can come from port C (PC[0:1,4:15, 23, 29]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	EDP C0	EDP C1	EDP C4	EDP C5	EDP C6	EDP C7	EDP C8	EDP C9	EDPC 10	EDPC 11	EDPC 12	EDPC 13	EDPC 14	EDPC 15	EDPC 23	EDPC 29
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10C24															

	16	17	18	19	20	21	22	23	24	31
Field	EDI0	EDI1	EDI2	EDI3	EDI4	EDI5	EDI6	EDI7	—	
Reset	0000_0000_0000_0000									
R/W	R/W								R	
Addr	0x10C26									

Figure 4-20. SIU External Interrupt Control Register (SIEXR)

Table 4-8 describes the SIEXR fields.

Table 4-8. SIEXR Field Descriptions

Bits	Name	Description
0–15	EDPCx	Edge detect mode for port Cx. The corresponding port C line (PCx) asserts an interrupt request according to the following: 0 Any change on PCx generates an interrupt request. 1 High-to-low change on PCx generates an interrupt request.
16–23	EDIx	Edge detect mode for $\overline{\text{IRQ}x}$. The corresponding IRQ line (IRQx) asserts an interrupt request according to the following: 0 Low assertion on $\overline{\text{IRQ}x}$ generates an interrupt request. 1 High-to-low change on $\overline{\text{IRQ}x}$ generates an interrupt request. **Note:** EDI5 applies both to IRQ5 and CINT.

4.3.2 System Configuration and Protection Registers

The system configuration and protection registers are described in the following sections.

4.3.2.1 Bus Configuration Register (BCR)

The bus configuration register (BCR), shown in Figure 4-21, contains configuration bits for various features and wait states on the 60x bus.

	0	1 3	4 7	8	9	10	11	12	13 15
Field	EBM	APD	—	PLDP	—	DAM	EAV	ETM	—
Reset	note 1	000_0000_0000_0000							
R/W	R/W								

	16 18	19 20	21	22 26	27	28 31
Field	NPQM	—	EXDD	—	ISPS	—
Reset	0000_0000_000				note 1	0000
R/W	R/W					
Addr	0x10024					

[1] Depends on reset configuration sequence. See Section 5.4.1, “Hard Reset Configuration Word.”

Figure 4-21. Bus Configuration Register (BCR)

Figure 4-9 describes the BCR fields.

Table 4-9. BCR Field Descriptions

Bits	Name	Description
0	EBM	External bus mode. 0 Single PowerQUICC II bus mode is assumed 1 60x-compatible bus mode. For more information refer to Section 8.2, “Bus Configuration.”
1–3	APD	Address phase delay. Specifies the minimum number of address tenure wait states for address operations initiated by a 60x bus master. BCR[APD] specifies the minimum number of address tenure wait states for address operations initiated by 60x-bus devices. APD indicates how many cycles the PowerQUICC II should wait for ARTRY, but because it is assumed that ARTRY can be asserted (by other masters) only on cacheable address spaces, APD is considered only on transactions that hit one of the 60x-assigned memory controller banks and have the GBL signal asserted during address phase.
4—7	—	Reserved, should be cleared.
8	PLDP	Pipeline maximum depth. See Section 8.4.5, “Pipeline Control.” 0 The pipeline maximum depth is one. 1 The pipeline maximum depth is zero.
9	—	Reserved, should be cleared.
10	DAM	Delay all masters. Applies to all the masters on the bus (CPU, EXT, CPM). This bit is similar to BCR[EXDD] but with opposite polarity. 0 The memory controller asserts CS on the cycle following the assertion of TS when accessing an address space controlled by the memory controller. 1 The memory controller inserts one wait state between the assertion of TS and the assertion of CS when accesses an address space controlled by the memory controller.
11	EAV	Enable address visibility. Normally, when the PowerQUICC II is in single-PowerQUICC II bus mode, the bank select signals for SDRAM accesses are multiplexed on the 60x bus address lines. So, for SDRAM accesses, the internal address is not visible for debug purposes. However the bank select signals can also be driven on dedicated pins (see SIUMCR[APPC]). In this case EAV can be used to force address visibility. 0 Bank select signals are driven on 60x bus address lines. There is no full address visibility. 1 Bank select signals are not driven on address bus. During READ and WRITE commands to SDRAM devices, the full address is driven on 60x bus address lines.

Table 4-9. BCR Field Descriptions (continued)

Bits	Name	Description
12	ETM	Compatibility mode enable. See Section 8.4.3.8, “Extended Transfer Mode.” 0 Strict 60x bus mode. Extended transfer mode is disabled. 1 Extended transfer mode is enabled.
13–15	—	Reserved, should be cleared.
16	NPQM0	Non PowerQUICC II master. Identifies the types of bus masters that are connected to the arbitration lines when the PowerQUICC II is in internal arbiter mode. Possible types are PowerQUICC II master and non-PowerQUICC II master. This field is related to the data pipelining bits (BRx[DR]) in the memory controller. Because an external bus master that is not a PowerQUICC II cannot use the data pipelining feature, the PowerQUICC II, which controls the memory, needs to know when a non-PowerQUICC II master is accessing the memory and handle the transaction differently. NPQM0 designates the type of master connected to the set of pins $\overline{\text{BR}}$, $\overline{\text{BG}}$, and $\overline{\text{DBG}}$. 0 The bus master connected to the arbitration lines is a PowerQUICC II. 1 The bus master connected to the arbitration lines is not a PowerQUICC II.
17	—	Reserved, should be cleared.
18	NPQM2	Non PowerQUICC II master. Identifies the type of bus masters that are connected to the arbitration lines when the PowerQUICC II is in internal arbiter mode. Possible types are PowerQUICC II master and non-PowerQUICC II master. This field is related to the data pipelining bits (BRx[DR]) in the memory controller. Because an external bus master that is not a PowerQUICC II cannot use the data pipelining feature, the PowerQUICC II, which controls the memory, needs to know when a non-PowerQUICC II master is accessing the memory and handle the transaction differently. NPQM0 designates the type of master that is connected to the set of pins EXT_BR3, EXT_BG3 and EXT_DBG3 0 The bus master connected to the arbitration lines is a PowerQUICC II. 1 The bus master connected to the arbitration lines is not a PowerQUICC II.
19–20	—	Reserved, should be cleared.
21	EXDD	External master delay disable. Generally, the MPC8280 adds one clock cycle delay for each external master access to a region controlled by the memory controller. This occurs because the external master drives the address on the external pins (compared to internal master, like MPC8280’s DMA, which drives the address on an internal bus in the chip). Thus, it is assumed that an additional cycle is needed for the memory controllers banks to complete the address match. However in some cases (when the bus is operated in low frequency), this extra cycle is not needed. The user can disable the extra cycle by setting EXDD.This bit is similar to BCR[DAM] but with opposite polarity. 0 The memory controller inserts one wait state between the assertion of $\overline{\text{TS}}$ and the assertion of CS when external master accesses an address space controlled by the memory controller. 1 The memory controller asserts CS on the cycle following the assertion of $\overline{\text{TS}}$ by external master accessing an address space controlled by the memory controller.
22–26	—	Reserved, should be cleared.
27	ISPS	Internal space port size. Defines the port size of PowerQUICC II’s internal space region as seen to external masters. Setting ISPS enables a 32-bit master to access PowerQUICC II internal space. 0 PowerQUICC II acts as a 64-bit slave to external masters accesses to its internal space. 1 PowerQUICC II acts as a 32-bit slave to external masters accesses to its internal space.
28–31	—	Reserved, should be cleared.

4.3.2.2 60x Bus Arbiter Configuration Register (PPC_ACR)

The 60x bus arbiter configuration register (PPC_ACR), shown in Figure 4-22, defines the arbiter modes and parked master on the 60x bus.

	0	1	2	3	4			7
Field	—		DBGD	EARB[1]	PRKM			
Reset	000			See note	0010			
R/W	R/W							
Addr	0x10028							

[1] Depends on reset configuration sequence. See Section 5.4.1, “Hard Reset Configuration Word.”

Figure 4-22. PPC_ACR

Table 4-10 describes the PPC_ACR fields.

Table 4-10. PPC_ACR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	DBGD	Data bus grant delay. Specifies the minimum number of data tenure wait states for 60x bus master-initiated data operations. This is the minimum delay between $\overline{\text{TS}}$ and $\overline{\text{DBG}}$. 0 $\overline{\text{DBG}}$ is asserted with $\overline{\text{TS}}$ if the data bus is free. 1 $\overline{\text{DBG}}$ is asserted one cycle after $\overline{\text{TS}}$ if the data bus is not busy. See Section 8.5.1, “Data Bus Arbitration.”
3	EARB	External arbitration. 0 Internal arbitration is performed. See Section 8.3.1, “Arbitration Phase.” 1 External arbitration is assumed.
4–7	PRKM	Parking master. 0000 CPM high request level refers to the IDMA, which involves peripherals and the following serial channels (SCC, SPI, SMC, and I^{2}C). 0001 CPM middle request level refers to all other serial channels (FCCs). 0010 CPM low request level: it is possible to change the request level for all FCCs to low priority when PPC_ACR[4–7] = 0010 and FCRx[1] = 1. (See Section 29.7.1, “FCC Function Code Registers (FCRx).” 0011 PCI request level. 0100 Security engine (SEC) 0101 Reserved 0110 Internal core 0111 External master 1 1000 Reserved 1001 External master 3 Values 1010–1111 are reserved.

4.3.2.3 60x Bus Arbitration-Level Registers (PPC_ALRH/PPC_ALRL)

The 60x bus arbitration-level registers, shown in Figure 4-23 and Figure 4-24, define arbitration priority of MPC8272 bus masters. Priority field 0 has highest-priority. For information about MPC8272 bus master indexes, see the description of PPC_ACR[PRKM] in Table 4-10.

	0 3	4 7	8 11	12 15
Field	Priority Field 0	Priority Field 1	Priority Field 2	Priority Field 3
Reset	0000	0001	0010	0110
R/W	R/W			
Addr	0x1002C			

	16 19	20 23	24 27	28 31
Field	Priority Field 4	Priority Field 5	Priority Field 6	Priority Field 7
Reset	0111	1000	1001	0011
R/W	R/W			
Addr	0x1002E			

Figure 4-23. PPC_ALRH

PPC_ALRL, shown in Figure 4-24, defines arbitration priority of 60x bus masters 8–15. Priority field 0 is the highest-priority arbitration level. For information about the MPC8272 bus master indexes, see the description of PPC_ACR[PRKM] in Table 4-10.

	0 3	4 7	8 11	12 15
Field	Priority Field 8	Priority Field 9	Priority Field 10	Priority Field 11
Reset	1000	1001	1010	1011
R/W	R/W			
Addr	0x10030			

	16 19	20 23	24 27	28 31
Field	Priority Field 12	Priority Field 13	Priority Field 14	Priority Field 15
Reset	1100	1101	1110	1111
R/W	R/W			
Addr	0x10032			

Figure 4-24. PPC_ALRL

4.3.2.4 SIU Module Configuration Register (SIUMCR)

The SIU module configuration register (SIUMCR), shown in Figure 4-25, contains bits that configure various features in the SIU module.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	BBD	ESE	—	CDIS	EXTMC		BAC		—		CPUC		CS6PC		BCTLC	
Reset	Note 1	00		See note 1					00		See note 1				00	
R/W	R/W															
Addr	0x10000															

	16	17	18	19	20	21	22	23	31
Field	MMR		—		DBE	ABE	SECDIS	—	
Reset	See note 1		00_0000_0000_0000						
R/W	R/W								
Addr	0x10002								

[1] Depends on reset configuration sequence. See Section 5.4.1, “Hard Reset Configuration Word.”

Figure 4-25. SIU Model Configuration Register (SIUMCR)

Table 4-11 describes the SIUMCR fields.

Table 4-11. SIUMCR Register Field Descriptions

Bits	Name	Description
0	BBD	Bus busy disable 0 $\overline{\text{ABB}}/\overline{\text{IRQ2}}$ pin is $\overline{\text{ABB}}$, $\overline{\text{DBB}}/\overline{\text{IRQ3}}$ pin is $\overline{\text{DBB}}$ 1 $\overline{\text{ABB}}/\overline{\text{IRQ2}}$ pin is $\overline{\text{IRQ2}}$, $\overline{\text{DBB}}/\overline{\text{IRQ3}}$ pin is $\overline{\text{IRQ3}}$
1	ESE	External snoop enable. Configures $\overline{\text{GBL}}/\overline{\text{IRQ1}}$ 0 External snooping disabled. ($\overline{\text{GBL}}/\overline{\text{IRQ1}}$ pin is $\overline{\text{IRQ1}}$) 1 External snooping enabled. ($\overline{\text{GBL}}/\overline{\text{IRQ1}}$ pin is $\overline{\text{GBL}}$)
2	—	Reserved, should be cleared.
3	CDIS	Core disable 0 The PowerQUICC II core is enabled. 1 The PowerQUICC II core is disabled. PowerQUICC II functions as a slave device.
4–5	EXTMC	External master pins configuration. Note that the additional arbitration lines (EXT_BR3, EXT_BG3, and EXT_DBG3) are operational only when ACR[EARB] = 0. Setting EARB (to choose external arbiter) combined with programming DPPC to 11 deactivates these lines.

Pin	EXTMC		
	00	10	11
$\overline{\text{IRQ3}}/\overline{\text{CKSTP\_OUT}}/\overline{\text{EXT\_BR3}}$	IRQ3	CKSTP_OUT	EXT_BR3
$\overline{\text{IRQ4}}/\overline{\text{CORE\_SRESET}}/\overline{\text{EXT\_BG3}}$	IRQ4	CORE_SRESET	EXT_BG3
$\overline{\text{IRQ5}}/\text{TBEN}/\overline{\text{EXT\_DBG3}}$	IRQ5	TBEN	EXT_DBG3

Table 4-11. SIUMCR Register Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>6–7</td><td>BAC</td><td>Burst address pins configuration.
<table>
<tr><th rowspan="2">Pin</th><th colspan="3">Multiplexing</th></tr>
<tr><th>BAC = 00</th><th>BAC = 01</th><th>BAC = 10</th></tr>
<tr><td>$\overline{CI}$/BADDR(29)/$\overline{IRQ2}$</td><td>CI</td><td>IRQ2</td><td>BADDR(29)</td></tr>
<tr><td>$\overline{WT}$/BADDR(30)/$\overline{IRQ3}$</td><td>WT</td><td>IRQ3</td><td>BADDR(30)</td></tr>
<tr><td>BADDR(31)/$\overline{IRQ5}$</td><td>—</td><td>IRQ5</td><td>BADDR(31)</td></tr>
</table></td></tr>
<tr><td>8–9</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>10–11</td><td>CPUC</td><td>CPU pins configuration. Note that during power on reset the MODCK pins are used for PLL configuration. The pin multiplexing indicated in the table applies only to normal operation.
<table>
<tr><th rowspan="2">Pin</th><th colspan="3">CPUC</th></tr>
<tr><th>00</th><th>01</th><th>10</th></tr>
<tr><td>MODCK1/$\overline{RSRV}$/TC(0)/ BNKSEL(0)</td><td>TC(0)</td><td>$\overline{RSRV}$</td><td>BNKSEL(0)</td></tr>
<tr><td>MODCK2/CSE0/TC(1)/ BNKSEL(1)</td><td>TC(1)</td><td>CSE0</td><td>BNKSEL(1)</td></tr>
<tr><td>MODCK3/CSE1/TC(2)/ BNKSEL(2)</td><td>TC(2)</td><td>CSE1</td><td>BNKSEL(2)</td></tr>
<tr><td>$\overline{CS7}$/$\overline{TLBISYNC}$</td><td>$\overline{CS7}$</td><td>$\overline{TLBISYNC}$</td><td>$\overline{CS7}$</td></tr>
</table></td></tr>
<tr><td>12–13</td><td>CS6PC</td><td>Chip select 6-pin configuration.
<table>
<tr><th rowspan="2">Pin</th><th colspan="3">CS6PC</th></tr>
<tr><th>00</th><th>01</th><th>10</th></tr>
<tr><td>$\overline{CS6}$/$\overline{BCTL1}$/$\overline{SMI}$</td><td>$\overline{CS6}$</td><td>$\overline{BCTL1}$</td><td>$\overline{SMI}$</td></tr>
</table></td></tr>
<tr><td>14–15</td><td>BCTLC</td><td>Configuration for the control lines for external buffers.
00 $\overline{BCTL0}$ is used as W/$\overline{R}$ control for external buffers. $\overline{BCTL1}$ is used as $\overline{OE}$ control for external buffers.
01 $\overline{BCTL0}$ is used as $\overline{W}$/R control for external buffers. $\overline{BCTL1}$ is used as $\overline{OE}$ control for external buffers.
10 $\overline{BCTL0}$ is used as $\overline{WE}$ control for external buffers. $\overline{BCTL1}$ is used as $\overline{RE}$ control for external buffers.
11 Reserved</td></tr>
</table>

Table 4-11. SIUMCR Register Field Descriptions (continued)

Bits	Name	Description
16–17	MMR	Mask masters requests. In some systems, several bus masters are active during normal operation; only one should be active during boot sequence. The active master, which is the boot device, initializes system memories and devices and enables all other masters. MMR facilitates such a boot scheme by masking the selected master's bus requests. MMR can be configured through the hard reset configuration sequence (see Section 5.4.2, "Hard Reset Configuration Examples"). Typically system configuration identifies only one master is the boot device, which initializes the system and enables all other devices by writing 00 to MMR. **Note:** It is not recommended to mask the request of a master that is defined as the parked master in the arbiter, since this cannot prevent this master from getting a bus grant. 00 No masking on bus request lines 01 Reserved 10 The PowerQUICC II's internal core bus request masked and external bus requests two and three masked (boot master connected to external bus request 1). 11 All external bus requests masked (boot master is the PowerQUICC II's internal core)
18–19	—	Reserved, should be cleared.
20	DBE	Data output buffer impedance configuration. The pins in this group include D[0:63] and $\overline{\text{PWE}}$[0:7]/$\overline{\text{PSDDQM}}$[0:7]/$\overline{\text{PBS}}$[0:7]. 0 The output buffer typical impedance is 45 Ω. 1 The output buffer typical impedance is 25 Ω.
21	ABE	Address output buffer impedance configuration. The pins in this group include A[0:31],PSDA10/PGPL0,$\overline{\text{PSDWE}}$/PGPL1,$\overline{\text{POE}}$/$\overline{\text{PSDRAS}}$/PGPL2,$\overline{\text{PSDCAS}}$/PGPL3,$\overline{\text{PGTA}}$/PUPM WAIT/pgpl4,PSDAMUX/PGPL5 and BNKSEL[0:2]. 0 The output buffer typical impedance is 45 Ω. 1 The output buffer typical impedance is 25 Ω.
22	SECDIS	Security co-processor disable. 0 The MPC8272 security co-processor is enabled. 1 The MPC8272 security co-processor is disabled.
23–31	—	Reserved, should be cleared.

4.3.2.5 Internal Memory Map Register (IMMR)

The internal memory map register (IMMR), shown in Figure 4-26, contains identification of a specific device as well as the base address for the internal memory map. Software can deduce availability and location of any on-chip system resources from the values in IMMR. PARTNUM and MASKNUM are mask programmed and cannot be changed for any particular device.

	0	13	14	15
Field	ISB		—	
Reset	Depends on reset configuration sequence. See Section 5.4.1, “Hard Reset Configuration Word.”			
R/W	R/W			
Addr	0x101A8			

	16	23	24	31
Field	PARTNUM		MASKNUM	
Reset	0x0C00 (MPC8272/MPC8241); 0x0D00 (MPC8271/MPC8247)			
R/W	R			
Addr	0x101AA			

Figure 4-26. Internal Memory Map Register (IMMR)

Table 4-12 describes the IMMR fields.

Table 4-12. IMMR Field Descriptions

Bits	Name	Description
0–13	ISB	Internal space base. Defines the base address of the internal memory space. The value of ISB is configured at reset to one of eight addresses; it can then be changed to any value by the software. The default is 0, which maps to address 0x0000_0000. (See Section 5.4.1, “Hard Reset Configuration Word.”) ISB defines the 14 msbs of the memory map register base address. IMMR itself is mapped in the internal memory space region. As soon as the ISB is written with a new base address, the IMMR base address is relocated according to the ISB. ISB can be configured to one of eight possible addresses at reset to enable the configuration of multiple-MPC8272 systems. The number of programmable bits in this field, and hence the resolution of the location of internal space, depends on the internal memory space of a specific implementation. In the MPC8272, all 14 bits can be programmed. See Chapter 3, “Memory Map,” for details on the device’s internal memory map and to Chapter 5, “Reset,” for the available default initial values.
14–15	—	Reserved, should be cleared.
16–23	PARTNUM	Part number. This read-only field is mask-programmed with a code corresponding to the part number of the part on which the SIU is located. It is intended to help factory test and user code that is sensitive to part changes. This changes when the part number changes. For example, it would change if any new module is added or if the size of any memory module is changed. It would not change if the part is changed to fix a bug in an existing module. The MPC8272 has an ID of 0x0C. The MPC8271/8247 has an ID of 0x0D.
24–31	MASKNUM	Mask number. This read-only field is mask-programmed with a code corresponding to the mask number of the part on which the SIU is located. It is intended to help factory test and user code that is sensitive to part changes. It is programmed in a commonly changed layer and should be changed for all mask set changes.

4.3.2.6 System Protection Control Register (SYPCR)

The system protection control register (SYPCR), shown in Figure 4-27, controls the system monitors, software watchdog period, and bus monitor timing. SYPCR can be read at any time but can be written only once after system reset.

	0	15
Field	SWTC	
Reset	1111_1111_1111_1111	
R/W	R/W	
Addr	0x10004	

	16	23	24	25	28	29	30	31
Field	BMT		PBME	—		SWE	SWRI	SWP
Reset	1111_1111		0	000_0		1	1	1
R/W	R/W							
Addr	0x10006							

Figure 4-27. System Protection Control Register (SYPCR)

Table 4-13 describes the SYPCR fields.

Table 4-13. SYPCR Field Descriptions

Bits	Name	Description
0–15	SWTC	Software watchdog timer count. Contains the count value for the software watchdog timer.
16–23	BMT	Bus monitor timing. Defines the time-out period for the bus monitor, the granularity of this field is eight bus clocks. (BMT = 0xFF is translated to 0x7f8 clock cycles). Note that the value 0 in invalid; an error is generated for each bus transaction.
24	PBME	60x bus monitor enable. 0 60x bus monitor is disabled. 1 The 60x bus monitor is enabled.
25–29	—	Reserved, should be cleared.
29	SWE	Software watchdog enable. Enables the operation of the software watchdog timer. It should be cleared by software after a system reset to disable the software watchdog timer.
30	SWRI	Software watchdog reset/interrupt select. 0 Software watchdog timer and bus monitor time-out cause a machine check interrupt to the core. 1 Software watchdog timer and bus monitor time-out cause a hard reset (this is the default value after soft reset).
31	SWP	Software watchdog prescale. Controls the divide-by-2,048 software watchdog timer prescaler. 0 The software watchdog timer is not prescaled. 1 The software watchdog timer clock is prescaled.

4.3.2.7 Software Service Register (SWSR)

The software service register (SWSR) is the location to which the software watchdog timer servicing sequence is written. To prevent software watchdog timer time-out, the user should write 0x556C followed by 0xAA39 to this register, which resides at 0x1000E. SWSR can be written at any time but returns all zeros when read.

4.3.2.8 60x Bus Transfer Error Status and Control Register 1 (TESCR1)

The 60x bus transfer error status and control register 1 (TESCR1) is shown in Figure 4-28.

	0	1	2 4	5	6	7 9	10	11 15
Field	BM	ISBE	—	WP	EXT	TC	—	TT
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x10040							

	16 18	19	20	21	22	23	24 31
Field	—	PCIMCP	—	IRQ0	SWD	ADO	—
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10042						

Note: Bits 0–15 and 19–23 are status bits and are cleared by writing 1s.

Figure 4-28. The 60x Bus Transfer Error Status and Control Register 1 (TESCR1)

Table 4-14 describes the TESCR1 fields.

Table 4-14. TESCR1 Field Descriptions

Bits	Name	Description
0	BM	60x bus monitor time-out. Set when $\overline{\text{TEA}}$ is asserted due to the 60x bus monitor time-out.
1	ISBE	Internal space bus error. Indicates that one of the following occurred: • $\overline{\text{TEA}}$ was asserted due to an error on a transaction to PowerQUICC II's internal memory space • An MCP was caused by a parity error on a transaction to PowerQUICC II's internal memory space. TESCR2[REGS, DPR, PCI0, PCI1] indicate which of PowerQUICC II's internal slaves caused the error.
2–4	—	Reserved, should be cleared.
5	WP	Write protect error. Indicates that a write was attempted to a 60x bus memory region that was defined as read-only in the memory controller. Note that this alone does not cause $\overline{\text{TEA}}$ assertion. Usually, in this case, the bus monitor will time-out.
6	EXT	External error. Indicates that $\overline{\text{TEA}}$ was asserted by an external bus slave.
7–9	TC	Transfer code. Indicates the transfer code of the 60x bus transaction that caused the $\overline{\text{TEA}}$ or MCP. See Section 8.4.3.2, "Transfer Code Signals TC[0:2]," for a description of the various transfer codes.
10	—	Reserved, should be cleared.
11–15	TT	Transfer type. These bits indicates the transfer type of the 60x bus transaction that caused the $\overline{\text{TEA}}$ or MCP. See Section 8.4.3.1, "Transfer Type Signal (TT[0:4]) Encoding," for a description of the various transfer types.
16–18	—	Reserved, should be cleared.
19	PCIMCP	PCI machine check. Set when a core machine check is asserted from the PCI bridge.
20	—	Reserved, should be cleared.

Table 4-14. TESCR1 Field Descriptions (continued)

Bits	Name	Description
21	IRQ0	External machine check. Set when a machine check is asserted due to the external machine check pin (IRQ0).
22	SWD	Software watchdog time-out. Indicates that a core machine check was asserted due to a time-out in the software watchdog. See Section 4.1.5, “Software Watchdog Timer.”
23	ADO	60x bus monitor address-only time-out. Set when a core machine check is asserted due to time-out of the bus monitor in an address only transaction. See Section 4.1.1, “Bus Monitor.”
24–31	—	Reserved, should be cleared.

4.3.2.9 60x Bus Transfer Error Status and Control Register 2 (TESCR2)

The 60x bus transfer error status and control register 2 (TESCR2) is shown in Figure 4-29.

<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6 … 15</td></tr>
<tr><td>Field</td><td>—</td><td>REGS</td><td>DPR</td><td>SEC</td><td>PCI0</td><td>PCI1</td><td>—</td></tr>
<tr><td>Reset</td><td colspan="7">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="7">R/W</td></tr>
<tr><td>Addr</td><td colspan="7">0x10044</td></tr>
</table>

<table>
<tr><td></td><td>16 … 31</td></tr>
<tr><td>Field</td><td>—</td></tr>
<tr><td>Reset</td><td>0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td>R/W</td></tr>
<tr><td>Addr</td><td>0x10046</td></tr>
</table>

Note: all bits are status bits and are cleared by writing 1s.

Figure 4-29. 60x Bus Transfer Error Status and Control Register 2 (TESCR2)

The TESCR2 register is described in Table 4-15.

Table 4-15. TESCR2 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	REGS	Internal registers error. An error occurred in a transaction to the PowerQUICC II’s internal registers.
2	DPR	Dual port ram error. An error occurred in a transaction to the PowerQUICC II’s dual-port RAM.
3	SEC	Security co-processor.
4	PCI0	PCI memory space 0 error. An error occurred in a transaction to the PCI memory space configured by PCIBR0 and PCIMSK0.
5	PCI1	PCI memory space 1 error. An error occurred in a transaction to the PCI memory space configured by PCIBR1 and PCIMSK1.
6–31	—	Reserved, should be cleared.

4.3.2.10 Time Counter Status and Control Register (TMCNTSC)

The time counter status and control register (TMCNTSC), shown in Figure 4-30, enables the different TMCNT functions and reports the source of the interrupts. The register can be read at any time. Status bits are cleared by writing ones; writing zeros does not affect the value of a status bit.

	0	7	8	9	10	11	12	13	14	15
Field	—		SEC	ALR	—		SIE	ALE	TCF	TCE
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	0x10220									

Figure 4-30. Time Counter Status and Control Register (TMCNTSC)

Table 4-16 describes the TMCNTSC fields.

Table 4-16. TMCNTSC Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	SEC	Once per second interrupt. This status bit is set every second and should be cleared by software.
9	ALR	Alarm interrupt. This status bit is set when the value of the TMCNT is equal to the value programmed in the alarm register.
10–11	—	Reserved, should be cleared.
12	SIE	Second interrupt enable. 0 The time counter does not generate an interrupt when SEC is set. 1 The time counter generates an interrupt when SEC is set.
13	ALE	Alarm interrupt enable. If ALE = 1, the time counter generates an interrupt when ALR is set.
14	TCF	Time counter frequency. The input clock to the time counter may be either 4 MHz or 32 KHz. The user should set the TCF bit according to the frequency of this clock. 0 The input clock to the time counter is 4 MHz. 1 The input clock to the time counter is 32 KHz. See Section 4.1.2, “Timers Clock,” for further details.
15	TCE	Time counter enable. Is not affected by soft or hard reset. 0 The time counter is disabled. 1 The time counter is enabled.

4.3.2.11 Time Counter Register (TMCNT)

The time counter register (TMCNT), shown in Figure 4-31, contains the current value of the time counter. The counter is reset to zero on $\overline{\text{PORESET}}$ reset or hard reset but is not affected by soft reset.

	0	15
Field	TMCNT	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10224	

	16	31
Field	TMCNT	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10226	

Figure 4-31. Time Counter Register (TCMCNT)

4.3.2.12 Time Counter Alarm Register (TMCNTAL)

The time counter alarm register (TMCNTAL), shown in Figure 4-32, holds a value (ALARM). When the value of TMCNT equals ALARM, a maskable interrupt is generated.

	0	15
Field	ALARM	
Reset	—	
R/W	R/W	
Addr	0x1022C	

	16	31
Field	ALARM	
Reset	—	
R/W	R/W	
Addr	0x1222E	

Figure 4-32. Time Counter Alarm Register (TMCNTAL)

Table 4-17 describes TMCNTAL fields.

Table 4-17. TMCNTAL Field Descriptions

Bits	Name	Description
0–31	ALARM	The alarm interrupt is generated when ALARM field matches the corresponding TMCNT bits. The resolution of the alarm is 1 second.

4.3.3 Periodic Interrupt Registers

The periodic interrupt registers are described in the following sections.

4.3.3.1 Periodic Interrupt Status and Control Register (PISCR)

The periodic interrupt status and control register (PISCR), shown in Figure 4-33, contains the interrupt request level and the interrupt status bit. It also contains the controls for the 16 bits to be loaded in a modulus counter.

	0 … 7	8	9 … 12	13	14	15
Field	—	PS	—	PIE	PTF	PTE
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x10240					

Figure 4-33. Periodic Interrupt Status and Control Register (PISCR)

Table 4-18 describes the PISCR fields.

Table 4-18. PISCR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	PS	Periodic interrupt status. Asserted if the PIT issues an interrupt. The PIT issues an interrupt after the modulus counter counts to zero. The PS bit can be negated by writing a one to PS. A write of zero has no effect on this bit.
9–12	—	Reserved, should be cleared.
13	PIE	Periodic interrupt enable. If PIE = 1, the periodic interrupt timer generates an interrupt when PS = 1.
14	PTF	Periodic interrupt frequency. The input clock to the periodic interrupt timer may be either 4 MHz or 32 KHz. The user should set the PTF bit according to the frequency of this clock. 0 The input clock to the periodic interrupt timer is 4 MHz. 1 The input clock to the periodic interrupt timer is 32 KHz. See Section 4.1.2, “Timers Clock,” for further details
15	PTE	Periodic timer enable. This bit controls the counting of the periodic interrupt timer. When the timer is disabled, it maintains its old value. When the counter is enabled, it continues counting using the previous value. 0 Disable counter 1 Enable counter

4.3.3.2 Periodic Interrupt Timer Count Register (PITC)

The periodic interrupt timer count register (PITC), shown in Figure 4-34, contains the 16 bits to be loaded in a modulus counter.

	0 … 15
Field	PITC
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10244

	16 … 31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10246

Figure 4-34. Periodic interrupt Timer Count Register (PITC)

Table 4-19 describes the PITC fields.

Table 4-19. PITC Field Descriptions

Bits	Name	Description
0–15	PITC	Periodic interrupt timing count. Bits 0–15 are defined as the PITC, which contains the count for the periodic timer. Setting PITC to 0xFFFF selects the maximum count period.
16–31	—	Reserved, should be cleared.

4.3.3.3 Periodic Interrupt Timer Register (PITR)

The periodic interrupt timer register (PITR), shown in Figure 4-35, is a read-only register that shows the current value in the periodic interrupt down counter. The PITR counter is not affected by reads or writes to it.

	0 … 15
Field	PIT
Reset	0000_0000_0000_0000
R/W	Read only
Addr	0x10248

	16 … 31
Field	—
Reset	0000_0000_0000_0000
R/W	Read only
Addr	0x1024A

Figure 4-35. Periodic Interrupt Timer Register (PITR)

Table 4-20 describes the PITR fields.

Table 4-20. PITR Field Descriptions

Bits	Name	Description
0–15	PITC	Periodic interrupt timing count. Bits 0–15 are defined as the PIT. It contains the current count remaining for the periodic timer. Writes have no effect on this field.
16–31	—	Reserved, should be cleared.

4.3.4 PCI Control Registers

Two pairs of registers detect accesses from the 60x bus side to the PCI bridge (other than PCI internal registers accesses). Each pair consists of a PCI base register (PCIBRx) for comparing addresses and a corresponding PCI mask register (PCIMSKx).

4.3.4.1 PCI Base Register (PCIBRx)

Figure 4-36 shows the PCI base register.

	0 15
Field	BA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x101AC (PCIBR0); 0x101B0 (PCIBR1)

	16	17 30	31
Field	BA	—	V
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x101AE (PCIBR0); 0x101B2 (PCIBR1)		

Figure 4-36. PCI Base Registers (PCIBRx)

Table 4-21 describes the PCIBRx fields.

Table 4-21. PCIBRx Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base address register are compared to the address on the 60x bus address bus to determine if the access should be claimed by the PCI bridge. Used with PCIMSKx[AM].
17–30	—	Reserved, should be cleared.
31	V	Valid bit. Indicates that the contents of the PCIBRx and PCIMSKx pairs are valid. 0 This pair is invalid. 1 This pair is valid.

4.3.4.2 PCI Mask Register (PCIMSKx)

Figure 4-37 shows the PCI mask register.

	0 — 15
Field	AM
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x101C4 (PCIBR0); 0x101C8 (PCIBR1)

	16	17 — 31
Field	AM	—
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x101C6 (PCIBR0); 0x101CA (PCIBR1)	

Figure 4-37. PCI Mask Register (PCIMSKx)

Table 4-22 describes the PCIMSKx fields.

Table 4-22. PCIMSKx Field Descriptions

Bits	Name	Description
0–16	AM	Address mask. Masks corresponding PCIBRx bits. 0 Corresponding address bits are masked. 1 Corresponding address bits are compared.
17–31	—	Reserved, should be cleared.

4.4 SIU Pin Multiplexing

Some functions share pins. The pinout of the MPC8272 is shown in the hardware specifications. The control of the actual functionality used on a specific pin is shown in Table 4-23.

Table 4-23. SIU Pins Multiplexing Control

Pin Name	Pin Configuration Control
GBL/IRQ1 CI/BADDR29/IRQ2 WT/BADDR30/IRQ3 BADDR31/IRQ5/CINT ABB/IRQ2 DBB/IRQ3 IRQ3/CKSTP_OUT/EXT_BR3 IRQ4/CORE_SRESET/EXT_BG3 IRQ5/TBEN/EXT_DBG3/CINT CS[6]/BCTL1/SMI CS[7]/TLBISYNC BNKSEL[0]/TC[0]/RSRV/MODCK1 BNKSEL[1]/TC[1]/CSE0/MODCK2 BNKSEL[2]/TC[2]/CSE1/MODCK3	Controlled by SIUMCR programming. See Section 4.3.2.4, "SIU Module Configuration Register (SIUMCR)," for more details.
PWE[0:7]/PSDDQM[0:7]/PBS[0:7] PSDA10/PGPL0 PSDWE/PGPL1 POE/PSDRAS/PGPL2 PSDCAS/PGPL3 PGTA/PUPMWAIT/PGPL4 PSDAMUX/PGPL5	Controlled dynamically according to the specific memory controller machine that handles the current bus transaction.

Chapter 5
Reset

The MPC8272 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
- Bus monitor reset
- Checkstop reset
- JTAG reset

All of these reset sources are fed into the reset controller and, depending on the source of the reset, different actions are taken. The reset status register, described in Section 5.2, “Reset Status Register (RSR),” indicates the last sources to cause a reset.

5.1 Reset Causes

Table 5-1 describes reset causes.

Table 5-1. Reset Causes

Name	Description
Power-on reset ($\overline{\text{PORESET}}$)	Input pin. Asserting this pin initiates the power-on reset flow that resets the chip and configures various attributes of the chip, including its clock mode.
Hard reset ($\overline{\text{HRESET}}$)	This is a bidirectional I/O pin. The MPC8272 can detect an external assertion of $\overline{\text{HRESET}}$ only if it occurs while the MPC8272 is not asserting reset. During $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ is asserted. $\overline{\text{HRESET}}$ is an open-collector pin.
Soft reset ($\overline{\text{SRESET}}$)	Bidirectional I/O pin. The MPC8272 can only detect an external assertion of $\overline{\text{SRESET}}$ if it occurs while the MPC8272 is not asserting reset. $\overline{\text{SRESET}}$ is an open-drain pin.
Software watchdog reset	After the MPC8272’s watchdog counts to zero, a software watchdog reset is signaled. The enabled software watchdog event then generates an internal hard reset sequence.
Bus monitor reset	After the MPC8272s bus monitor counts to zero, 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.

5.1.1 Reset Actions

The reset block has a reset control logic that determines the cause of a reset, synchronizes it if necessary, and resets the appropriate logic modules. The memory controller, system protection logic, interrupt controller, and parallel I/O pins are initialized only on hard reset. Soft reset initializes the internal logic while maintaining the system configuration. Because there is no soft or hard reset in the 603e core, asserting external SRESET generates a reset to the 603e core and a soft reset to the remainder of the device. The reset to the core resets the MSR[IP] to the value in the HRCW[CIP]; see Table 5-7.

Table 5-2 identifies reset actions for each reset source.

Table 5-2. Reset Actions for Each Reset Source

Reset Source	Reset Logic and PLL States Reset	System Configuration Sampled	Clock Module Reset	HRESET Driven	Other Internal Logic Reset[1]	SRESET Driven	Core Reset
Power-on reset	Yes	Yes	Yes	Yes	Yes	Yes	Yes
External hard reset Software watchdog Bus monitor Checkstop	No	Yes	Yes	Yes	Yes	Yes	Yes
JTAG reset External soft reset	No	No	No	No	Yes	Yes	Yes

[1] Includes all other CPM and core logic not explicitly noted elsewhere in the table.

5.1.2 Power-On Reset Flow

Assertion of the PORESET external pin initiates the power-on reset flow. PORESET should be asserted externally for at least 16 input clock cycles after external power to the chip reaches at least 2/3 Vcc. The value driven on RSTCONF while PORESET changes from assertion to negation determines the chip configuration. If RSTCONF is negated (driven high) while PORESET changes, the chip acts as a configuration slave. If RSTCONF is asserted while PORESET changes, the chip acts as a configuration master. Section 5.4, “Reset Configuration,” explains the configuration sequence and the terms ‘configuration master’ and ‘configuration slave.’

Directly after the negation of PORESET and choice of the reset operation mode as configuration master or configuration slave, the MPC8272 starts the configuration process. The MPC8272 asserts HRESET and SRESET throughout the power-on reset process, including during configuration. Configuration takes 1,024 CLOCKIN cycles, after which MODCK[1:3] are sampled to determine the chips working mode. Next, the MPC8272 halts until the main PLL locks. As described in Section 10.6, “Clock Configuration Modes,” the main PLL locks according to MODCK[1:3], which are sampled, and to MODCK_HI (MODCK[4:7]), taken from the reset configuration word. During this time HRESET and SRESET are asserted. When the main PLL is locked, the clock block starts distributing clock signals in the chip. HRESET remains asserted for another 512 clocks before being released. The SRESET is released three clocks later.

Figure 5-4 shows the power-on reset flow.

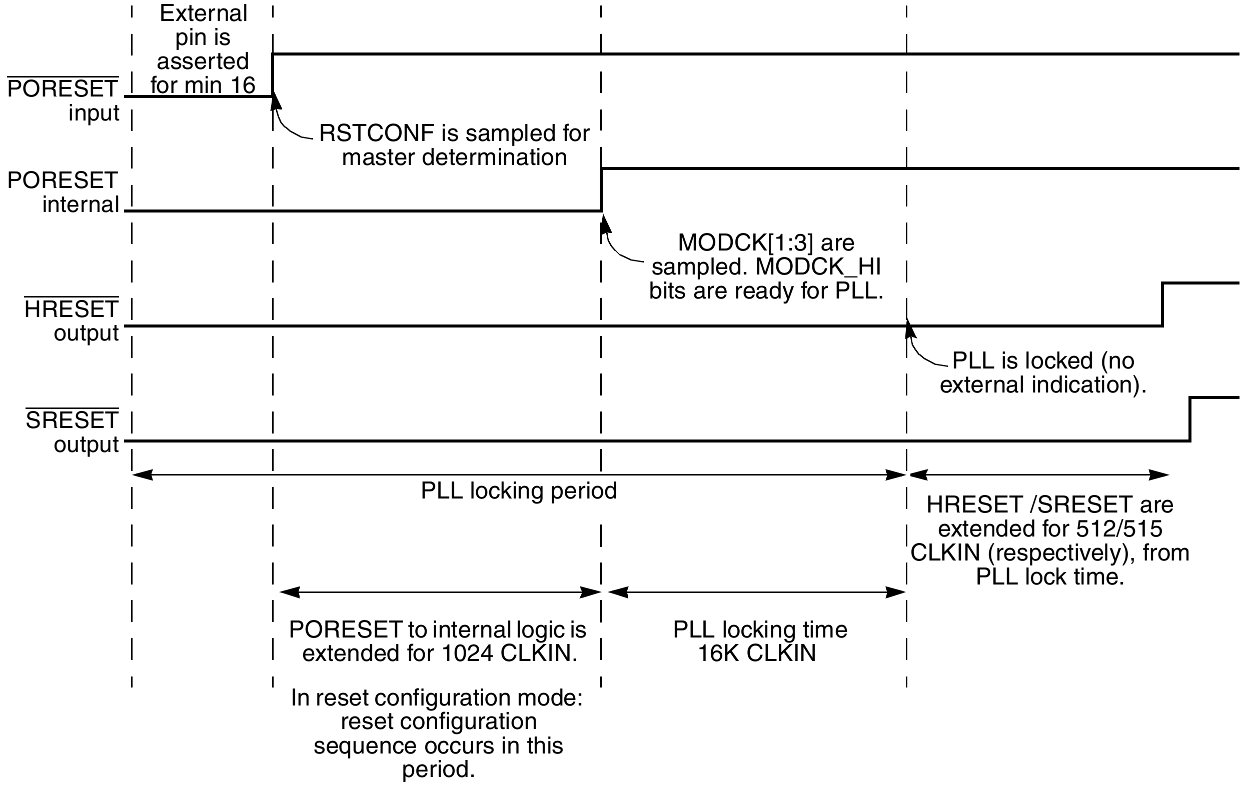

External pin is asserted for min 16
PORESET input
RSTCONF is sampled for master determination
PORESET internal
MODCK[1:3] are sampled. MODCK_HI bits are ready for PLL.
HRESET output
PLL is locked (no external indication).
SRESET output
PLL locking period
HRESET /SRESET are extended for 512/515 CLKIN (respectively), from PLL lock time.
PORESET to internal logic is extended for 1024 CLKIN.
PLL locking time 16K CLKIN
In reset configuration mode: reset configuration sequence occurs in this period.

Figure 5-1. Power-on Reset Flow

5.1.3 HRESET Flow

The HRESET flow may be initiated externally by asserting HRESET or internally when the chip detects a reason to assert HRESET. In both cases the chip continues asserting HRESET and SRESET throughout the HRESET flow. The HRESET flow begins with the hard reset configuration sequence, which configures the chip as explained in Section 5.4, “Reset Configuration.” After the chip asserts HRESET and SRESET for 1,024 input clock cycles, it releases both signals and exits the HRESET flow. An external pull-up resistor should negate the signals. After negation is detected, a 16-cycle period is taken before testing the presence of an external (hard/soft) reset.

5.1.4 SRESET Flow

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

5.2 Reset Status Register (RSR)

The reset status register (RSR), shown in Figure 5-2, is memory-mapped into the MPC8272’s SIU register map.

	0–15
Field	—
R/W	R/W
Reset	0000_0000_0000_0000
Addr	0x10C90

	16–25	26	27	28	29	30	31
Field	—	JTRS	CSRS	SWRS	BMRS	ESRS	EHRS
R/W	R/W						
Reset	0000_0000_0000_0011						
Addr	0x10C92						

Figure 5-2. Reset Status Register (RSR)

Table 5-3 describes RSR fields.

Table 5-3. RSR Field Descriptions

Bits	Name	Function
0–25	—	Reserved, should be cleared.
26	JTRS	JTAG reset status. When the JTAG reset request is set, JTRS is set and remains set until software clears it. JTRS 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.
27	CSRS	Checkstop reset status. When the core enters a checkstop state and the checkstop reset is enabled by the RMR[CSRE], CSRS is set and remains set until software clears it. CSRS is cleared by writing a 1 to it (writing zero has no effect). 0 No enabled checkstop reset event occurred. 1 An enabled checkstop reset event occurred.
28	SWRS	Software watchdog reset status. When a software watchdog expire event (which causes a reset) is detected, the SWRS bit is set and remains that way until the software clears it. SWRS is cleared by writing a 1 to it (writing zero has no effect). 0 No software watchdog reset event 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

Table 5-3. RSR Field Descriptions (continued)

Bits	Name	Function
30	ESRS	External soft reset status. When an external soft reset event is detected, ESRS is set and it remains that way until software clears it. ESRS is cleared by writing a 1 to it (writing zero has no effect). 0 No external soft reset event has occurred 1 An external soft reset event has occurred
31	EHRS	External hard reset status. When an external hard reset event is detected, EHRS is set and it remains set until software clears it. EHRS is cleared by writing a 1 (writing zero has no effect). 0 No external hard reset event has occurred 1 An external hard reset event has occurred

NOTE

The reset status register (RSR) 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[ESRS] and RSR[EHRS] are all set after a software watchdog reset.

5.3 Reset Mode Register (RMR)

The reset mode register (RMR), shown in Figure 5-3, is memory-mapped into the SIU register map.

	0 … 15
Field	—
R/W	R/W
Reset	0000_0000_0000_0000
Addr	0x10C94

	16 … 30	31
Field	—	CSRE
R/W	R/W	
Reset	0000_0000_0000_0000	
Addr	0x10C96	

Figure 5-3. Reset Mode Register (RMR)

Table 5-4 describes RMR fields.

Table 5-4. 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.

5.4 Reset Configuration

Various features may be configured during hard reset or power-on reset. For example, one configurable feature is core disable, which can be used to configure a system that uses two MPC8272s—one as a slave device and the other as the host with an active core. Most configurable features are reconfigured whenever HRESET is asserted. However, the clock mode is configured only when PORESET is asserted.

The 32-bit hard reset configuration word is described in Section 5.4.1, "Hard Reset Configuration Word." The reset configuration sequence is designed to support a system that uses up to eight MPC8272 chips, each configured differently. It needs no additional glue logic for reset configuration.

The description below explains the operation of this sequence with regard to a multiple-MPC8272 system. This and other simpler systems are described in Section 5.4.2, "Hard Reset Configuration Examples." In a typical multi-MPC8272 system, one MPC8272 should act as the configuration master while all other MPC8272s should act as configuration slaves. The configuration master in the system typically reads the various configuration words from EEPROM in the system and uses them to configure itself as well as the configuration slaves. The value of the RSTCONF input determines how the MPC8272 acts during reset configuration, while PORESET changes from assertion to negation. If RSTCONF is asserted while PORESET changes, the MPC8272 is a configuration master; otherwise, it is a slave.

In a typical multiple-MPC8272 system, RSTCONF input of the configuration master should be hard wired to ground, while RSTCONF inputs of other chips should be connected to the high-order address bits of the configuration master, as described in Table 5-5.

Table 5-5. RSTCONF Connections in Multiple-MPC8272 Systems

Configured Device	RSTCONF Connection
Configuration master	GND
First configuration slave	A0
Second configuration slave	A1
Third configuration slave	A2
Fourth configuration slave	A3
Fifth configuration slave	A4
Sixth configuration slave	A5
Seventh configuration slave	A6

The configuration words for all MPC8272s are assumed to reside in an EEPROM connected to CS0 of the configuration master. Because the port size of this EEPROM is not known to the configuration master, before reading the configuration words, the configuration master reads all configuration words byte-by-byte only from locations that are independent of port size.

Table 5-6. shows addresses that should be used to configure the various MPC8272s. Byte addresses that do not appear in this table have no effect on the configuration of the MPC8272 chips. The values of the bytes in Table 5-6 are always read on byte lane D[0–7] regardless of the port size.

Table 5-6. Configuration EEPROM Addresses

Configured Device	Byte 0 Address	Byte 1 Address	Byte 2 Address	Byte 3 Address
Configuration master	0x00	0x08	0x10	0x18
First configuration slave	0x20	0x28	0x30	0x38
Second configuration slave	0x40	0x48	0x50	0x58
Third configuration slave	0x60	0x68	0x70	0x78
Fourth configuration slave	0x80	0x88	0x90	0x98
Fifth configuration slave	0xA0	0xA8	0xB0	0xB8
Sixth configuration slave	0xC0	0xC8	0xD0	0xD8
Seventh configuration slave	0xE0	0xE8	0xF0	0xF8

The configuration master first reads a value from address 0x00 and then reads a value from addresses 0x08, 0x10, and 0x18. These four bytes are used to form the configuration word of the configuration master, which then proceeds to read the bytes that form the configuration word of the first slave device. The configuration master drives the whole configuration word on D[0:31] and toggles its A0 address line. Each configuration slave uses its $\overline{\text{RSTCONF}}$ input as a strobe for latching the configuration word during $\overline{\text{HRESET}}$ assertion time. Thus, the first configuration slave whose $\overline{\text{RSTCONF}}$ input is connected to configuration master's A0 output latches the word driven on D[0:31] as its configuration word. In this way the configuration master continues to configure all MPC8272 chips in the system. The configuration master always reads eight configuration words regardless of the number of MPC8272 parts in the system. In a simple system that uses one standalone MPC8272, it is possible to use the default hard reset configuration word (all zeros). This is done by tying the $\overline{\text{RSTCONF}}$ input to VCC. Another scenario may be a system that has no boot EEPROM. In this case the user can configure the MPC8272 as a configuration slave by driving $\overline{\text{RSTCONF}}$ to 1 during $\overline{\text{PORESET}}$ assertion and applying a negative pulse on $\overline{\text{RSTCONF}}$ and an appropriate configuration word on D[0:31]. In such a system, asserting $\overline{\text{HRESET}}$ in the middle of operation causes the MPC8272 to return to the configuration programmed after $\overline{\text{PORESET}}$ assertion (not the default configuration represented by configuration word of all zeros).

5.4.1 Hard Reset Configuration Word

The contents of the hard reset configuration word are shown in Figure 5-4.

	0	1	2	3	4 5	6	7	8 9	10 11	12	13 15
Field	EARB	EXMC	CDIS	EBM	BPS	CIP	ISPS	BAC	EXTMC	PLLBP	ISB
Reset	0000_0000_0000_0000										

	16	17	18 19	20 21	22 23	24 25	26	27	28 31
Field	BMS	BBD	MMR	—	CPUC	CS6PC	ALD_EN	PCI_MODCK	MODCK_H
Reset	0000_0000_0000_0000								

Figure 5-4. Hard Reset Configuration Word

Table 5-7 describes hard reset configuration word fields.

Table 5-7. Hard Reset Configuration Word Field Descriptions

Bits	Name	Description
0	EARB[1]	External arbitration. Defines the initial value for ACR[EARB]. If EARB = 1, external arbitration is assumed. See Section 4.3.2.2, “60x Bus Arbiter Configuration Register (PPC_ACR).”
1	EXMC	External MEMC. Defines the initial value of BR0[EMEMC]. If EXMC = 1, an external memory controller is assumed. See Section 11.3.1, “Base Registers (BRx).”
2	CDIS[1]	Core disable. Defines the initial value for the SIUMCR[CDIS]. 0 The core is active. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).” 1 The core is disabled. In this mode the MPC8272 functions as a slave.
3	EBM[1]	External bus mode. Defines the initial value of BCR[EBM]. See Section 4.3.2.1, “Bus Configuration Register (BCR).”
4–5	BPS	Boot port size. Defines the initial value of BR0[PS], the port size for memory controller bank 0. 00 64-bit port size 01 8-bit port size 10 16-bit port size 11 32-bit port size See Section 11.3.1, “Base Registers (BRx).”
6	CIP[1]	Core initial prefix. Defines the initial value of MSR[IP]. Exception prefix. The setting of this bit specifies whether an exception vector offset is prepended with Fs or 0s. In the following description, *nnnnn* is the offset of the exception vector. 0 MSR[IP] = 1 (default). Exceptions are vectored to the physical address 0xFFF*n_nnnn* 1 MSR[IP] = 0 Exceptions are vectored to the physical address 0x000*n_nnnn*.
7	ISPS[1]	Internal space port size. Defines the initial value of BCR[ISPS]. Setting ISPS configures the MPC8272 to respond to accesses from a 32-bit external master to its internal space. See Section 4.3.2.1, “Bus Configuration Register (BCR).”
8–9	BAC[1]	Burst address pins configuration. Defines the initial value of SIUMCR[BAC]. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).”
10–11	EXTMC[1]	External master pin configuration. Defines the initial value of SIUMCR[EXTMC]. For more details refer to Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).”
12	PLLBP	Reset configuration bypass PLL. 0 Normal operation 1 Bypass CPM PLL The effect of setting this bit occurs only after $\overline{\text{PORESET}}$ is asserted.
13–15	ISB	Initial internal space base select. Defines the initial value of IMMR[0–13] and determines the base address of the internal memory space. 000 0x0000_0000 001 0x00F0_0000 010 0x0F00_0000 011 0x0FF0_0000 100 0xF000_0000 101 0xF0F0_0000 110 0xFF00_0000 111 0xFFF0_0000 See Section 4.3.2.5, “Internal Memory Map Register (IMMR).”

Table 5-7. Hard Reset Configuration Word Field Descriptions (continued)

Bits	Name	Description
16	BMS	Boot memory space. Defines the initial value for BR0[BA]. There are two possible boot memory regions: HIMEM and LOMEM. 0 0xFE00_0000—0xFFFF_FFFF 1 0x0000_0000—0x01FF_FFFF See Section 11.3.1, “Base Registers (BRx).”
17	BBD[1]	Bus busy disable. Defines the initial value of SIUMCR[BBD]. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).”
18–19	MMR	Mask masters requests. Defines the initial value of SIUMCR[MMR]. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).”
20–21	—	Reserved, should be cleared.
22–23	CPUC[1]	CPU pins configuration. Defines the initial value of SIUMCR[CPUC]. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).”
24–25	CS6PC[1]	CS6 pin configuration. Defines the initial value of SIUMCR[CS6PC]. See Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR).” **Note:** During the reset configuration sequence, the BCTL1/CS6 pin toggles like $\overline{\text{POE}}$ of the 60x bus GPCM, regardless of the configuration of the reset configuration word. After the reset configuration sequence, the BCTL1/CS6 pin behaves according to the configuration of SIUMCR[CS6PC].
26	ALD_EN	CP auto load enable. Allows the CP to automatically load the essential PCI configuration registers from the EEPROM during reset. 0 CP auto load is disabled. 1 CP auto load is enabled.
27	PCI_ MODCK	Reset configuration pci_modck. Determines PCI clock frequency range. Refer to “Clock Configuration Mode,” in the *MPC8272 Family Hardware Specifications* (MPC8272EC). 0 PCI high frequency range 1 PCI low frequency range
28–31	MODCK_H	High-order bits of the MODCK bus, which determine the clock reset configuration. See Chapter 10, “PLL and Clock Generator,” for details.

[1] The user should exercise caution when changing this bit. This bit has an immediate effect on the external bus and may result in unstable system operation.

5.4.2 Hard Reset Configuration Examples

This section presents some examples of hard reset configurations in different systems.

5.4.2.1 Single MPC8272 with Default Configuration

This is the simplest configuration scenario. It can be used if the default values achieved by clearing the hard reset configuration word are desired. This is applicable only for systems using single-MPC8272 bus mode (as opposed to 60x bus mode). To enter this mode, tie $\overline{\text{RSTCONF}}$ to V_{CC} as shown in Figure 5-5. The MPC8272 does not access the boot EEPROM; it is assumed that the default configuration is used upon exiting hard reset.

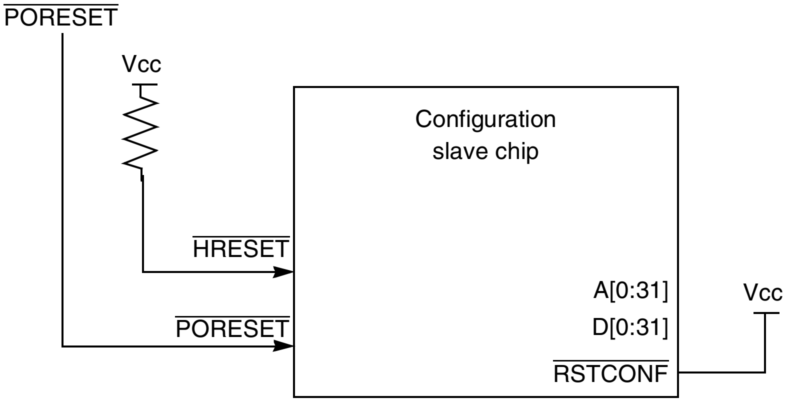


Figure 5-5. Single Chip with Default Configuration

5.4.2.2 Single MPC8272 Configured from Boot EEPROM

For a configuration that differs from the default, the MPC8272 can be used as a configuration master by tying RSTCONF to GND as shown in Figure 5-6. The MPC8272 can access the boot EEPROM. It is assumed the configuration is as defined there upon exiting hard reset.

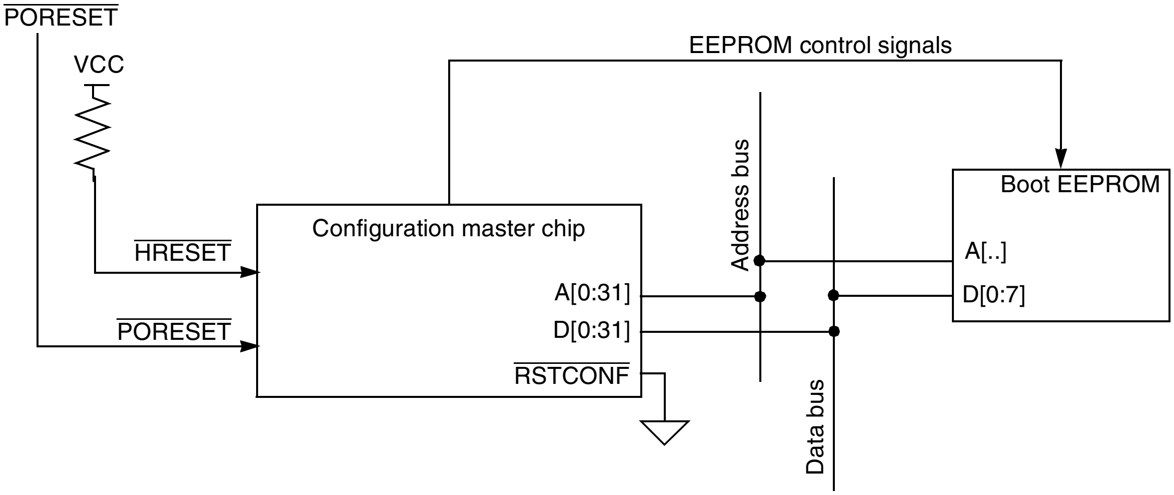


Figure 5-6. Configuring a Single Chip from EEPROM

5.4.2.3 Multiple MPC8272s Configured from Boot EEPROM

For a complex system with multiple MPC8272 devices that may each be configured differently, configuration is done by assigning one configuration master and multiple configuration slaves. The MPC8272 that controls the boot EEPROM should be the configuration master—RSTCONF tied to GND. The RSTCONF inputs of the other MPC8272 devices are tied to the address bus lines, thus assigning them as configuration slaves. See Figure 5-7.

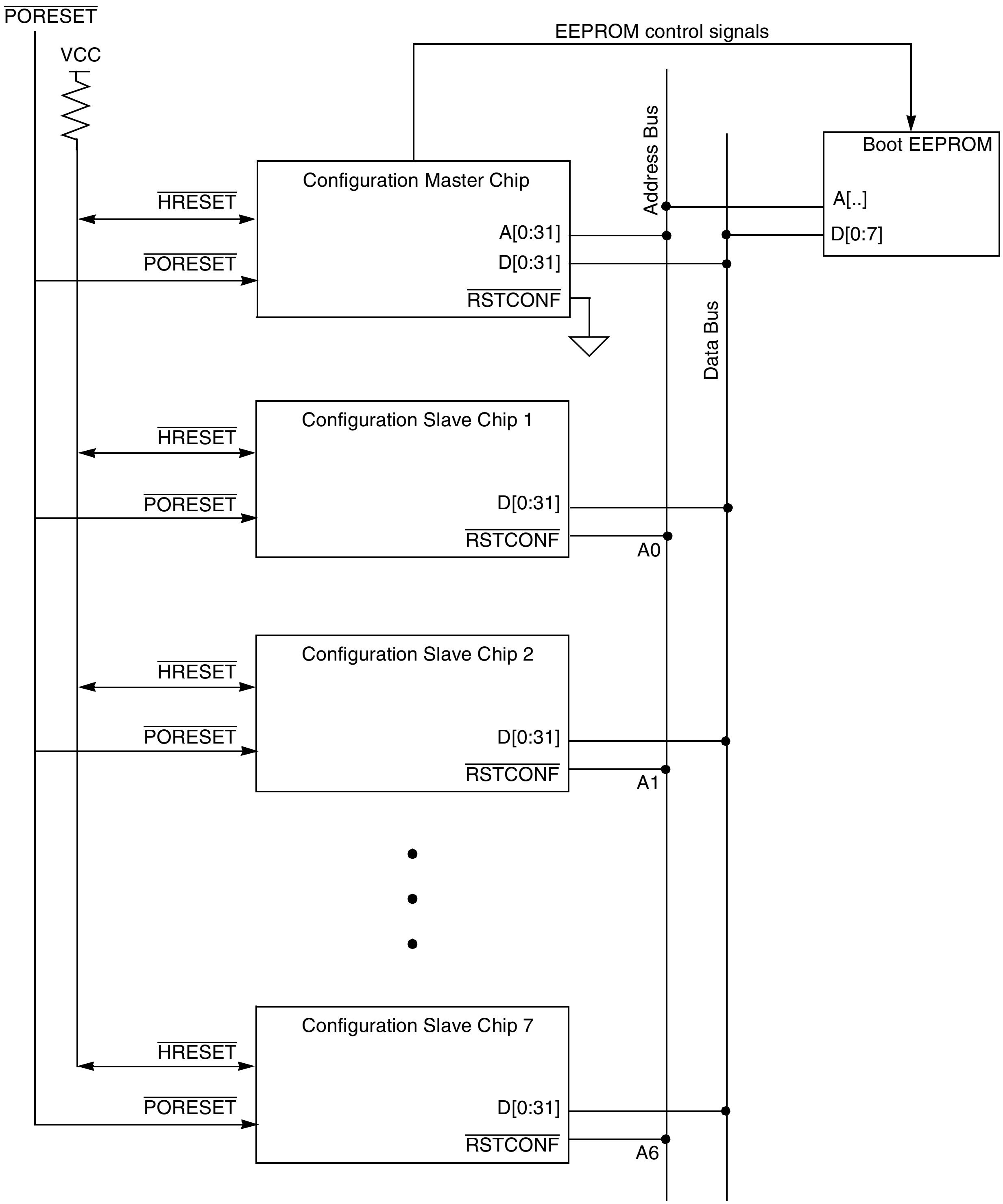


Figure 5-7. Configuring Multiple Chips

In this system, the configuration master initially reads its own configuration word. It then reads other configuration words and drives them to the configuration slaves by asserting $\overline{\text{RSTCONF}}$. As Figure 5-7 shows, this complex configuration is done without additional glue logic. The configuration master controls the whole process by asserting the EEPROM control signals and the system's address signals as needed.

5.4.2.4 Multiple MPC8272s in a System with No EEPROM

In some cases, the configuration master capabilities of the MPC8272 cannot be used. This can happen for example if there is no boot EEPROM in the system or the boot EEPROM is not controlled by a MPC8272.

If this occurs, the user must do one of the following:

- Accept the default configuration.
- Emulate the configuration master actions in external logic (where the MPC8272 is a configuration slave).
- The external hardware should be connected to all $\overline{\text{RSTCONF}}$ pins of the different devices and to the upper 32 bits of the data bus. During $\overline{\text{PORESET}}$, the rising edge the external hardware should negate all $\overline{\text{RSTCONF}}$ inputs to put all of the devices in their configuration slave mode. For 1,024 clocks after $\overline{\text{PORESET}}$ negation, the external hardware can configure the different devices by driving appropriate configuration words on the data bus and asserting $\overline{\text{RSTCONF}}$ for each device to strobe the data being received.

Part III
The Hardware Interface

Intended Audience

This part is intended for system designers who need to understand how each MCP8272 signal works and how those signals interact.

Contents

This part describes external signals, clocking, memory control, and power management in the MPC8272.

It contains the following chapters:

- Chapter 6, "External Signals," shows a functional pinout of the MPC8272 and describes the MPC8272 signals.
- Chapter 7, "60x Signals," describes signals on the 60x bus.
- Chapter 8, "The 60x Bus," describes the operation of the bus used by PowerPC processors.
- Chapter 9, "PCI Bridge," describes how the PCI bridge enables the MPC8272 to gluelessly bridge PCI agents to a host processor that implements the PowerPC architecture and how it is compliant with PCI Specification Revision 2.2.
- Chapter 10, "PLL and Clock Generator," describes the clocking architecture of the MPC8272.
- Chapter 11, "Memory Controller," describes the memory controller, which controls a maximum of eight memory banks shared between a general-purpose chip-select machine (GPCM) and three user-programmable machines (UPMs).
- Chapter 12, "IEEE 1149.1 Test Access Port," describes the dedicated user-accessible test access port (TAP), which is fully compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture*.

Suggested Reading

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

MPC82xx Documentation

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.freescale.com. This documentation includes technical specifications, reference materials, and detailed applications notes.

Conventions

This document uses the following notational conventions:

Bold — Bold entries in figures and tables showing registers and parameter RAM should be initialized by the user.

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.

0x0 — Prefix to denote hexadecimal number

0b0 — Prefix to denote binary number

REG[FIELD] — Abbreviations or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.

x — In certain contexts, such as in a signal encoding or a bit field, indicates a don’t care.

n — Indicates an undefined numerical value

¬ — NOT logical operator

& — AND logical operator

| — OR logical operator

Acronyms and Abbreviations

Table III-1 contains acronyms and abbreviations used in this document. Note that the meanings for some acronyms (such as SDR1 and DSISR) are historical, and the words for which an acronym stands may not be intuitively obvious.

Table III-1. Acronyms and Abbreviated Terms

Term	Meaning
BD	Buffer descriptor
BIST	Built-in self test
BRI	Basic rate interface
CAM	Content-addressable memory
CPM	Communications processor module
CRC	Cyclic redundancy check
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory

Table III-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
DSISR	Register used for determining the source of a DSI exception
EA	Effective address
EEST	Enhanced Ethernet serial transceiver
GCI	General circuit interface
GPCM	General-purpose chip-select machine
HDLC	High-level data link control
I^2C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IrDA	Infrared Data Association
ISDN	Integrated services digital network
JTAG	Joint Test Action Group
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
MSR	Machine state register
NMSI	Nonmultiplexed serial interface
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PCMCIA	Personal Computer Memory Card International Association
PRI	Primary rate interface
Rx	Receive
SCC	Serial communications controller
SCP	Serial control port
SDLC	Synchronous data link control

Table III-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
SDMA	Serial DMA
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
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
UART	Universal asynchronous receiver/transmitter
UISA	User instruction set architecture
UPM	User-programmable machine
USART	Universal synchronous/asynchronous receiver/transmitter

Chapter 6
External Signals

This chapter describes the MPC8272's external signals.

6.1 Functional Pinout

Figure 6-1 shows signals grouped by function. Note that many signals are multiplexed and this figure does not indicate how they are multiplexed.

NOTE

A bar over a signal name indicates that the signal is active low—for example, $\overline{\text{BR}}$ (bus request). 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 ALE (address latch enable), are referred to as asserted when they are high and negated when they are low.

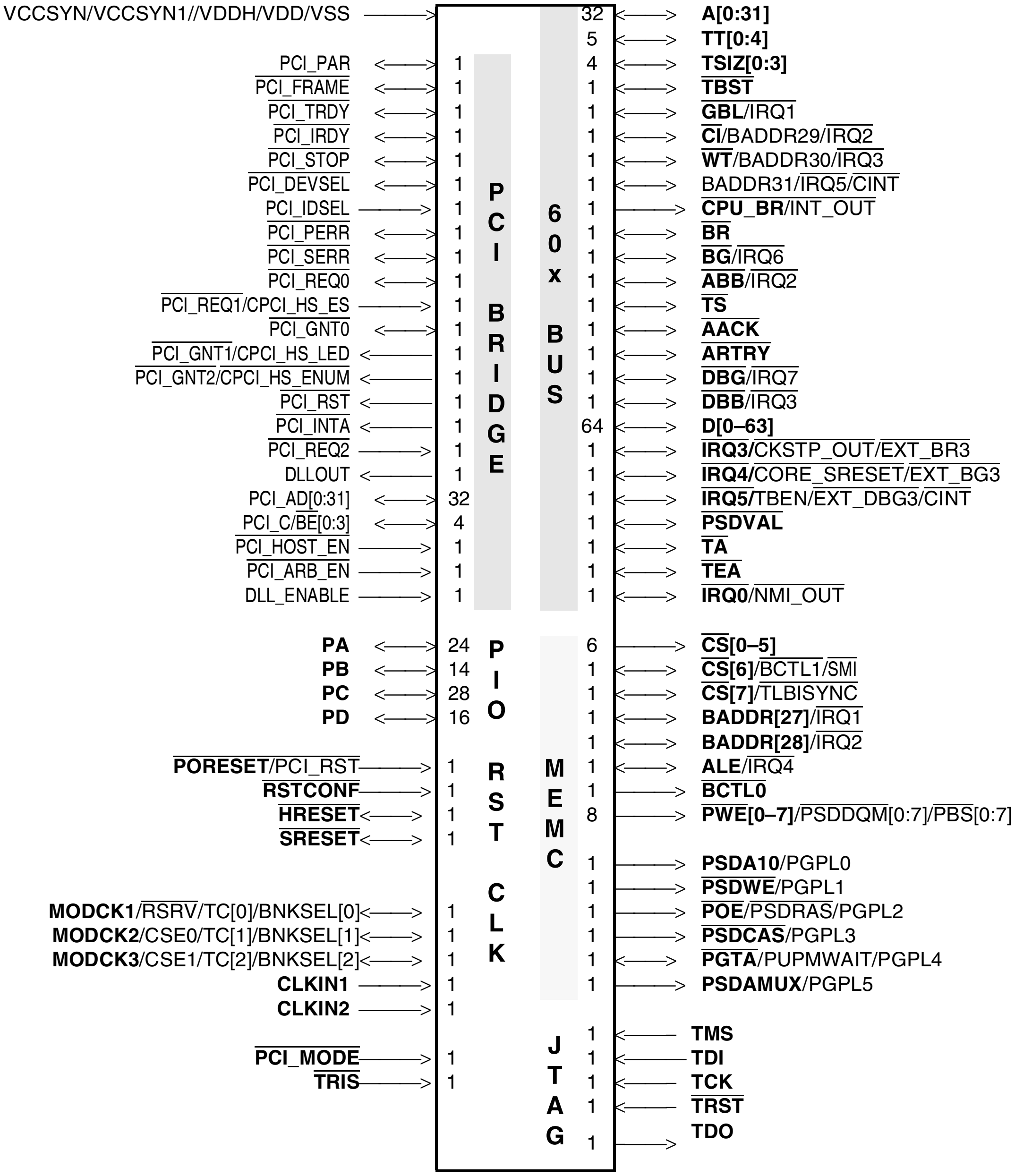


Figure 6-1. External Signals

6.2 Signal Descriptions

The system bus consists of all the signals that interface with the external bus. These signals are described in Table 6-1. Many of these pins perform different functions, depending on how the user assigns them.

Table 6-1. External Signals

Signal	Description
$\overline{\text{BR}}$	60x bus request—This is an output when an external arbiter is used and an input when an internal arbiter is used. As an output the MPC8272 asserts this pin to request ownership of the 60x bus. As an input an external master should assert this pin to request 60x bus ownership from the internal arbiter.
$\overline{\text{BG}}$ $\overline{\text{IRQ6}}$	60x bus grant—This is an output when an internal arbiter is used and an input when an external arbiter is used. As an output the MPC8272 asserts this pin to grant 60x bus ownership to an external bus master. As an input the external arbiter should assert this pin to grant 60x bus ownership to the MPC8272. Interrupt request 6—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
$\overline{\text{ABB}}$ $\overline{\text{IRQ2}}$	60x address bus busy—(Input/output) As an output the MPC8272 asserts this pin for the duration of the address bus tenure. Following an $\overline{\text{AACK}}$, which terminates the address bus tenure, the MPC8272 negates $\overline{\text{ABB}}$ for a fraction of a bus cycle and stops driving this pin. As an input the MPC8272 will not assume 60x bus ownership as long as it senses this pin is asserted by an external 60x bus master. Interrupt request 2—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
$\overline{\text{TS}}$	60x bus transfer start—(Input/output) Assertion of this pin signals the beginning of a new address bus tenure. The MPC8272 asserts this signal when one of its internal 60x bus masters (core, DMA, PCI bridge) begins an address tenure. When the MPC8272 senses this pin being asserted by an external 60x bus master, it will respond to the address bus tenure as required (snoop if enabled, access internal MPC8272 resources, memory controller support).
A[0:31]	60x address bus—These are input/output pins. When the MPC8272 is in external master bus mode, these pins function as the 60x address bus. The MPC8272 drives the address of its internal 60x bus masters and respond to addresses generated by external 60x bus masters. When the MPC8272 is in internal master bus mode, these pins are used as address lines connected to memory devices and controlled by the MPC8272's memory controller.
TT[0:4]	60x bus transfer type—These are input/output pins. The 60x bus master drives these pins during the address tenure to specify the type of the transaction.
$\overline{\text{TBST}}$	60x bus transfer burst—(Input/output) The 60x bus master asserts this pin to indicate that the current transaction is a burst transaction (transfers 4 double words).
TSIZ[0:3]	60x transfer size—These are input/output pins. The 60x bus master drives these pins with a value indicating the amount of bytes transferred in the current transaction.
$\overline{\text{AACK}}$	60x address acknowledge—This is an input/output signal. A 60x bus slave asserts this signal to indicate that it identified the address tenure. Assertion of this signal terminates the address tenure.
$\overline{\text{ARTRY}}$	60x address retry—(Input/output) Assertion of this signal indicates that the bus transaction should be retried by the 60x bus master. The MPC8272 asserts this signal to enforce data coherency with its internal cache and to prevent deadlock situations.

Table 6-1. External Signals (continued)

Signal	Description
DBG IRQ7	60x data bus grant—This is an output when an internal arbiter is used and an input when an external arbiter is used. As an output the MPC8272 asserts this pin to grant 60x data bus ownership to an external bus master. As an input the external arbiter should assert this pin to grant 60x data bus ownership to the MPC8272. Interrupt Request 7—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
DBB IRQ3	60x data bus busy—(Input/output) As an output the MPC8272 asserts this pin for the duration of the data bus tenure. Following a TA, which terminates the data bus tenure, the MPC8272 negates DBB for a fraction of a bus cycle and than stops driving this pin. As an input, the MPC8272 does not assume 60x data bus ownership as long as it senses DBB asserted by an external 60x bus master. Interrupt request 3—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
D[0:63]	60x data bus—These are input/output pins. In write transactions the 60x bus master drives the valid data on this bus. In read transactions the 60x slave drives the valid data on this bus.
IRQ3 CKSTP_OUT EXT_BR3	Interrupt request 3—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core. Checkstop output—(Output) Assertion indicates that the core is in its checkstop mode. External bus request 3—(Input) An external master should assert this pin to request 60x bus ownership from the internal arbiter.
IRQ4 CORE_SRESET EXT_BG3	Interrupt request 4—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core. Core system reset—(Input) Asserting this pin will force the core to branch to its reset vector. External bus grant 3—(Output) The MPC8272 asserts this pin to grant 60x bus ownership to an external bus master.
IRQ5 TBEN EXT_DBG3 CINT	Interrupt request 5—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core. Time base enable—This is a count enable input to the time base counter in the core. External data bus grant 3—(Output) The MPC8272 asserts this pin to grant 60x data bus ownership to an external bus master. Critical interrupt—Critical interrupt input to the core
PSDVAL	60x data valid—(Input/output) Assertion of the PSDVAL pin indicates that a data beat is valid on the data bus. The difference between the TA pin and the PSDVAL pin is that the TA pin is asserted to indicate 60x data transfer termination while the PSDVAL signal is asserted with each data beat movement. Thus, whenever TA is asserted, PSDVAL will be asserted but when PSDVAL is asserted, TA is not necessarily asserted. For example when a double word (2x64-bit) transfer is initiated by the SDMA to a memory device that has a 32-bit port size, PSDVAL will be asserted three times without TA and both pins will be asserted to terminate the transfer.

Table 6-1. External Signals (continued)

Signal	Description
$\overline{\text{TA}}$	Transfer acknowledge—(Input/output) Indicates that a 60x data beat is valid on the data bus. For 60x single beat transfers, assertion of this pin indicates the termination of the transfer. For 60x burst transfers $\overline{\text{TA}}$ is asserted four times to indicate the transfer of four data beats with the last assertion indicating the termination of the burst transfer.
$\overline{\text{TEA}}$	Transfer error acknowledge—(Input/output) Assertion of this pin indicates a bus error. 60x masters within the MPC8272 monitor the state of this pin. The MPC8272's internal bus monitor may assert this pin in case it identified a 60x bus transfer that is hung.
$\overline{\text{GBL}}$ $\overline{\text{IRQ1}}$	Global—(Input/output) When a 60x master within the chip initiates a bus transaction it drives this pin. When an external 60x master initiates a bus transaction it should drive this pin. Assertion of this pin indicates that the transfer is global and it should be snooped by caches in the system. The MPC8272's data cache monitors the state of this pin. Interrupt request 1—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
$\overline{\text{CI}}$ BADDR29 $\overline{\text{IRQ2}}$	Cache inhibit—Output pin. For each MPC8272 60x transaction initiated in the core, the state of this pin indicates if this transaction was cached or not. Assertion of the $\overline{\text{CI}}$ pin indicates that the transaction was not cached. Burst address 29—There are five burst address output pins. These pins are outputs of the 60x memory controller. These pins are used in external master configuration and are connected directly to memory devices controlled by the MPC8272's memory controller. Interrupt request 2—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
$\overline{\text{WT}}$ BADDR30 $\overline{\text{IRQ3}}$	Write through—Output pin. For each core-initiated PowerQUICC II 60x transaction, the state of this pin indicates if the transaction was cached using write-through or copy-back mode. Assertion of $\overline{\text{WT}}$ indicates that the transaction was cached using the write-through mode. Burst address 30—There are five burst address output pins. These pins are outputs of the 60x memory controller. These pins are used in external master configuration and are connected directly to memory devices controlled by the MPC8272's memory controller. Interrupt request 3—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
BADDR31 $\overline{\text{IRQ5}}$ $\overline{\text{CINT}}$	Burst address 31—There are five burst address output of the 60x memory controller used in an external master configuration and are connected directly to the memory devices controlled by the MPC8272's memory controller. Interrupt request 5—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core. Critical interrupt—Critical interrupt input to the core.
$\overline{\text{CPU\_BR}}$ $\overline{\text{INT\_OUT}}$	CPU bus request—(Output) The value of the 60x core bus request is driven on this pin. Interrupt output—This is an output driven from the MPC8272's internal interrupt controller. Assertion of this output indicates that an unmasked interrupt is pending in the MPC8272's internal interrupt controller.
$\overline{\text{CS}}$[0:5]	Chip select—These are output pins that enable specific memory devices or peripherals connected to the MPC8272 buses.

Table 6-1. External Signals (continued)

Signal	Description
$\overline{\text{CS}}$[6] $\overline{\text{BCTL1}}$ $\overline{\text{SMI}}$	Chip select—Output pins that enable specific memory devices or peripherals connected to the MPC8272 buses. Buffer control 1—Output signal whose function is controlling buffers on the 60x data bus. Usually used with $\overline{\text{BCTL0}}$. The exact function of this pin is defined by the value of SIUMCR[BCTLC]. System management interrupt—System management interrupt input to the core.
$\overline{\text{CS}}$[7] $\overline{\text{TLBISYNC}}$	Chip select—Output that enable specific memory devices or peripherals connected to PowerQUICC II buses. TLB sync—Input pin that can be used to synchronize 60x core instruction execution to hardware indications. Asserting this pin will force the core to stop instruction execution following a **tlbsync** instruction execution. The core resumes instructions execution once this pin is negated.
BADDR[27] $\overline{\text{IRQ1}}$	Burst address 27—There are five burst address output pins. These pins are outputs of the 60x memory controller. Used in external master configuration and connected directly to the memory devices controlled by the MPC8272's memory controller. Interrupt request 1—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
BADDR[28] $\overline{\text{IRQ2}}$	Burst address 28—There are five burst address output pins. These pins are outputs of the 60x memory controller. Used in external master configuration and connected directly to the memory devices controlled by the MPC8272's memory controller. Interrupt request 2—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
ALE $\overline{\text{IRQ4}}$	Address latch enable—This output pin controls the external address latch that should be used in external master 60x bus configuration. Interrupt Request 4—This input is one of the eight external lines that can request (by means of the internal interrupt controller) a service routine from the core.
$\overline{\text{BCTL0}}$	Buffer control 0—Output whose function is controlling buffers on the 60x data bus. Usually used with $\overline{\text{BCTL1}}$ that is multiplexed on $\overline{\text{CS6}}$. The exact function of this pin is defined by the value of SIUMCR[BCTLC].
$\overline{\text{PWE}}$[0:7] $\overline{\text{PSDDQM}}$[0:7] PBS[0:7]	60x bus write enable—Outputs of the 60x bus GPCM. These pins select byte lanes for write operations. 60x bus SDRAM DQM—Outputs of the SDRAM control machine. These pins select specific byte lanes of SDRAM devices. 60x bus UPM byte select—Outputs of the UPM in the memory controller. They are used to select specific byte lanes during memory operations. The timing of these pins is programmed in the UPM. The actual driven value depends on the address and size of the transaction and the port size of the accessed device.
PSDA10 PGPL0	60x bus SDRAM A10—(Output) from the 60x bus SDRAM controller. Part of the address when a row address is driven and is part of the command when a column address is driven. 60x bus UPM general-purpose line 0—One of six general-purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.

Table 6-1. External Signals (continued)

Signal	Description
$\overline{\text{PSDWE}}$ PGPL1	60x bus SDRAM write enable—Output from the 60x bus SDRAM controller. Should be connected to SDRAMs' WE input. 60x bus UPM general purpose line 1—One of six general purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.
$\overline{\text{POE}}$ $\overline{\text{PSDRAS}}$ PGPL2	60x bus output enable—Output of the 60x bus GPCM. Controls the output buffer of memory devices during read operations. 60x bus SDRAM ras—Output from the 60x bus SDRAM controller. Should be connected to SDRAMs' RAS input. 60x bus UPM general purpose line 2—One of six general purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.
$\overline{\text{PSDCAS}}$ PGPL3	60x bus SDRAM CAS—Output from the 60x bus SDRAM controller. Should be connected to SDRAMs' CAS input. 60x bus UPM general purpose line 3—One of six general purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.
$\overline{\text{PGTA}}$ PUPMWAIT PGPL4	60x GPCM TA—Input pin used for transaction termination during GPCM operation. Requires external pull up resistor for proper operation. 60x bus UPM wait—This is an input to the UPM. An external device may hold this pin high to force the UPM to wait until the device is ready for the continuation of the operation. 60x bus UPM general purpose line 4—One of six general purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.
PSDAMUX PGPL5	60x bus SDRAM address multiplexer—This output pin controls the 60x SDRAM address multiplexer when the MPC8272 is in external master mode. 60x bus UPM general purpose line 5—One of six general purpose output lines from UPM. The values and timing of this pin is programmed in the UPM.
$\overline{\text{PCI\_HOST\_EN}}$	PCI host enable—This is an input to the PCI. When high, enables the PCI bridge for agent operation; when low, enables the PCI as host.
$\overline{\text{PCI\_ARB\_EN}}$	PCI arbiter enable—An input to the PCI. When low, enables the PCI internal arbiter logic; when high, disables the internal arbiter logic (and an external arbiter should be used).
DLL_ENABLE	DLL enable—This input pin should be pulled high externally in order to use the DLL.
PCI_PAR	PCI parity—PCI parity input/output pin. Assertion of this pin indicates that odd parity is driven across PCI_AD[31:0] and PCI_C/$\overline{\text{BE}}$[3:0] during address and data phases. Negation of PCI_PAR indicates that even parity is driven across the PCI_AD[31:0] and PCI_C/$\overline{\text{BE}}$[3:0] during address and data phases.
$\overline{\text{PCI\_FRAME}}$	PCI frame—PCI cycle frame input/output pin. Used by the current PCI master to indicate the beginning and duration of an access. Driven by the MPC8272 when its PCI interface is the master of the access. Otherwise, it is an input.
$\overline{\text{PCI\_TRDY}}$	PCI target ready—PCI target ready input/output pin. This pin is driven by the MPC8272 when its PCI interface is the target of a PCI transfer. Assertion of this pin indicates that the PCI target is ready to send or accept a data beat.

Table 6-1. External Signals (continued)

Signal	Description
$\overline{\text{PCI\_IRDY}}$	PCI initiator ready—PCI initiator ready input/output pin. This pin is driven by the MPC8272 when its PCI interface is the initiator of a PCI transfer. Assertion of this pin indicates that the PCI initiator is ready to send or accept a data beat.
$\overline{\text{PCI\_STOP}}$	PCI stop—PCI stop input/output pin. This pin is driven by the MPC8272 when its PCI interface is the target of a PCI transfer. Assertion of this pin indicates that the PCI target is requesting the master to stop the current PCI transfer.
$\overline{\text{PCI\_DEVSEL}}$	PCI device select—PCI device select input/output pin. This pin is driven by the MPC8272 when its PCI interface has decoded its own address as the target of the current PCI transfer. As an input, $\overline{\text{PCI\_DEVSEL}}$ indicates whether any device on the PCI bus has been selected.
PCI_IDSEL	PCI initialization device select—Input used to select the MPC8272's PCI interface during a PCI configuration cycle.
$\overline{\text{PCI\_PERR}}$	PCI parity error—PCI data parity error input/output pin. Assertion of this pin indicates that a data parity error was detected during a PCI transfer (except for a special cycle).
$\overline{\text{PCI\_SERR}}$	PCI system error—PCI system error input/output pin. Assertion of this pin indicates that a PCI system error was detected during a PCI transfer. The PCI system error is for reporting address parity errors, data parity errors on a special cycle command, or other catastrophic system errors.
$\overline{\text{PCI\_REQ0}}$	PCI arbiter request 0—PCI request 0 input/output pin. When the MPC8272's internal PCI arbiter is used, this is an input pin. In this mode assertion of this pin indicates that an external PCI device is requesting the PCI bus. When an external PCI arbiter is used, this is an output pin. In this mode assertion of this pin indicates that the MPC8272's PCI interface is requesting the PCI bus.
$\overline{\text{PCI\_REQ1}}$ CPCI_HS_ES	PCI arbiter request 1—PCI request 1 input pin. When the MPC8272's internal PCI arbiter is used, assertion of this pin indicates that an external PCI device is requesting the PCI bus. CompactPCI Hot Swap ejector switch—Hot Swap ejector switch input pin. In a CompactPCI system, when the MPC8272's internal PCI arbiter is not used, this pin is used for the Hot Swap interface to connect to the ejector switch logic. 0 Switch is closed 1 Switch is open Important note: When functioning as the CPCI_HS_ES input, this signal must be filtered (debounced) by an external circuit. Do not connect this input directly to the ejector switch. The input must be a monotonically rising/falling signal.
$\overline{\text{PCI\_GNT0}}$	PCI arbiter grant 0—PCI grant 0 input/output pin. When the MPC8272's internal PCI arbiter is used, this is an output pin. In this mode, assertion of $\overline{\text{PCI\_GNT0}}$ indicates that an the external PCI device that requested the PCI bus with $\overline{\text{PCI\_REQ0}}$ is granted the bus. When an external PCI arbiter is used, this is an input pin. In this mode, assertion of $\overline{\text{PCI\_GNT0}}$ indicates that the MPC8272's PCI interface is granted the PCI bus.
$\overline{\text{PCI\_GNT1}}$ CPCI_HS_LED	PCI arbiter grant 1—PCI grant 1 output pin. When the MPC8272's internal PCI arbiter is used, assertion of $\overline{\text{PCI\_GNT1}}$ indicates that the external PCI device that requested the PCI bus with $\overline{\text{PCI\_REQ1}}$ pin is granted the bus. CompactPCI Hot Swap LED—Hot Swap LED output pin. In CompactPCI system, when the MPC8272's internal PCI arbiter is not used, this pin is used for the Hot Swap interface to connect to the Hot Swap LED. The Hot Swap pins are not available when the internal arbiter is used. 0 LED is off. 1 LED is on.

Table 6-1. External Signals (continued)

Signal	Description
PCI_GNT2 CPCI_HS_ENUM	PCI arbiter grant 2—PCI grant 2 output pin. When the MPC8272's internal PCI arbiter is used, assertion of PCI_GNT2 indicates that the external PCI device that requested the PCI bus with PCI_REQ2 pin is granted the bus. CompactPCI Hot Swap enumerator—Hot Swap ENUM output pin. In CompactPCI system, when the MPC8272's internal PCI arbiter is not used, this pin is used for the Hot Swap interface to connect to the host as the enumeration request.
PCI_RST	PCI reset—PCI reset output pin. When the MPC8272 is the host in the PCI system, PCI_RST is an output.
PCI_INTA	PCI INTA—(output) When the MPC8272 is an agent of the PCI system, this pin is an output used by the MPC8272 to signal an interrupt to the PCI host. (When the MPC8272 is the host in the PCI system, the general IRQ pins are used for delivering PCI interrupts to the host.)
PCI_REQ2	PCI arbiter request 2—PCI request 2 input pin. When the MPC8272's internal PCI arbiter is used, assertion of this pin indicates that an external PCI device is requesting the PCI bus.
DLLOUT	DLL clock out—DLL output pin. This is the DLL output reference clock.
PCI_AD[31-0]	PCI address/data—PCI bus address/data input/output pins. During an address phase PCI_AD[31-0] contains a physical address, during a data phase PCI_AD[31-0] contains the data bytes. In the PCI address/data bus, bit 31 is msb and bit 0 is lsb.
PCI_C/BE[3-0]	PCI command/byte enable—PCI command/byte enable input/output pins. The MPC8272 drives these pins when it is the initiator of a PCI transfer. During an address phase the PCI_C/BE[3-0] defines the command, during the data phase PCI_C/BE[3-0] defines the byte enables. PCI_C/BE[3] is the msb and PCI_C/BE[0] is the lsb.
IRQ0 NMI_OUT	Interrupt request 0—This input is an external line that causes an MCP interrupt to the core. Non-maskable interrupt output—An output driven from the MPC8272's internal interrupt controller. Assertion of this output indicates that a non-maskable interrupt is pending in the MPC8272's internal interrupt controller.
TRST	Test reset (JTAG)— Input only. This is the reset input to the MPC8272's JTAG/COP controller.
TCK	Test clock (JTAG)—Input only. Provides the clock input for the MPC8272's JTAG/COP controller.
TMS	Test mode select (JTAG)—Input only. Controls the state of the MPC8272's JTAG/COP controller.
TDI	Test data in (JTAG)—Input only. Data input to the MPC8272's JTAG/COP controller.
TDO	Test data out (JTAG)—Output only. Data output from the MPC8272's JTAG/COP controller.
TRIS	Three-state—Asserting TRIS forces all other MPC8272's pins to high impedance state.
PORESET PCI_RST	Power-on reset—When asserted, this input line causes the MPC8272 to enter power-on reset state. PCI reset—PCI reset input pin. When the MPC8272 is an agent in the PCI system, PCI_RST is an input.
HRESET	Hard reset—This open drain line, when asserted causes the MPC8272 to enter hard reset state.
SRESET	Soft reset—This open drain line, when asserted causes the MPC8272 to enter the soft reset state.
RSTCONF	RSTCONF —Input used during reset configuration sequence of the chip.

Table 6-1. External Signals (continued)

Signal	Description
MODCK1 $\overline{\text{RSRV}}$ TC[0] BNKSEL[0]	MODCK1—Clock mode input. Defines the operating mode of internal clock circuits. Reservation—The value driven on this output pin represents the state of the coherency bit in the reservation address register that is used by the **lwarx** and **stwcx.** instructions. Transfer code 0—The transfer code output pins supply information that can be useful for debug purposes for each of the MPC8272's initiated bus transactions. Bank select 0—The bank select outputs are used for selecting SDRAM bank when the MPC8272 is in 60x compatible bus mode. BNKSEL0 is msb of the three BNKSEL signals.
MODCK2 CSE[0] TC[1] BNKSEL[1]	MODCK2—Clock mode input. Defines the operating mode of internal clock circuits. Cache set entry 0—The cache set entry outputs from the core represent the cache replacement set element for the current core transaction reloading into or writing out of the cache. Transfer code 1—The transfer code output pins supply information that can be useful for debug purposes for each of the MPC8272's initiated bus transactions. Bank select 1—The bank select outputs are used for selecting SDRAM bank when the MPC8272 is in 60x-compatible bus mode.
MODCK3 CSE[1] TC[2] BNKSEL[2]	MODCK3—Clock mode input. Defines the operating mode of internal clock circuits. Cache set entry 1—The cache set entry outputs from the core represent the cache replacement set element for the current core transaction reloading into or writing out of the cache. Transfer code 2—The transfer code output pins supply information that can be useful for debug purposes for each of the MPC8272's initiated bus transactions. Bank select 2—The bank select outputs are used for selecting SDRAM bank when the MPC8272 is in 60x-compatible bus mode. BNKSEL2 is lsb of the three BNKSEL signals.
CLKIN1	Clock In1—Primary clock input to MPC8272's PLL. When the MPC8272 is an agent in the PCI system, CLKIN1 should be connected to the PCI bus clock. In that case, the 60x bus clock is driven on CLKOUT.
CLKIN2	Clock In2—This is the clock input to the MPC8272's DLL, which is used for de-skewing the output reference clock.
$\overline{\text{PCI\_MODE}}$	PCI mode pin - This pin enables the PCI bridge of the MPC8272. This pin must by tied to ground at any time
PA	General-purpose I/O port A—CPM port multiplexing is described in Chapter 37, "Parallel I/O Ports."
PB	General-purpose I/O port B—CPM port multiplexing is described in Chapter 37, "Parallel I/O Ports."
PC	General-purpose I/O port C—CPM port multiplexing is described in Chapter 37, "Parallel I/O Ports."
PD	General-purpose I/O port D—CPM port multiplexing is described in Chapter 37, "Parallel I/O Ports."
Power supply	VDD—This is the power supply of the internal logic. VDDH—This is the power supply of the I/O buffers. VCCSYN—This is the power supply of the PLL circuitry. VCCSYN1—This is the power supply of the core's PLL circuitry.

Chapter 7
60x Signals

This chapter describes the signals on the 60x bus. It contains a concise description of individual signals, showing behavior when a signal is asserted and negated, when the signal is an input and an output, and differences in how signals work in external-master or internal-only configurations.

NOTE

> A bar over a signal name indicates that the signal is active low– for example, $\overline{\text{ARTRY}}$ (address retry) and $\overline{\text{TS}}$ (transfer start). 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 TSIZ[0:3] (transfer size signals) and TT[0:4] (transfer type signals), are referred to as asserted when they are high and negated when they are low.

The 60x bus signals used with MPC8272 are grouped as follows:

- Address arbitration signals—In external arbiter mode, MPC8272 uses these signals to arbitrate for address bus mastership. The MPC8272 arbiter uses these signals to enable an external device to arbitrate for address bus mastership.
- Address transfer start signals—These signals indicate that a bus master has begun a transaction on the address bus.
- Address transfer signals (address bus)—These signals are used to transfer the address.
- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is single, single extended, bursted, write-through or cache-inhibited.
- Address transfer termination signals—These signals are used to acknowledge the end of the address phase of the transaction. They also indicate whether a condition exists that requires the address phase to be repeated.
- Data arbitration signals—The MPC8272, in external arbiter mode, uses these signals to arbitrate for data bus mastership. The MPC8272 arbiter uses these signals to enable an external device to arbitrate for data bus mastership.
- Data transfer signals—Data bus signals transfer the data.
- Data transfer termination signals—Data termination signals are required after each data beat in a data transfer. In a single-beat transaction, the data termination signals also indicate the end of the tenure. For burst accesses or extended port-size accesses, the data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat.

7.1 Signal Configuration

Figure 7-1 shows the grouping of the MPC8272's 60x bus signal configuration.

NOTE

> The MPC8272 hardware specifications provides a pinout showing pin numbers. These are shown in Figure 7-1.

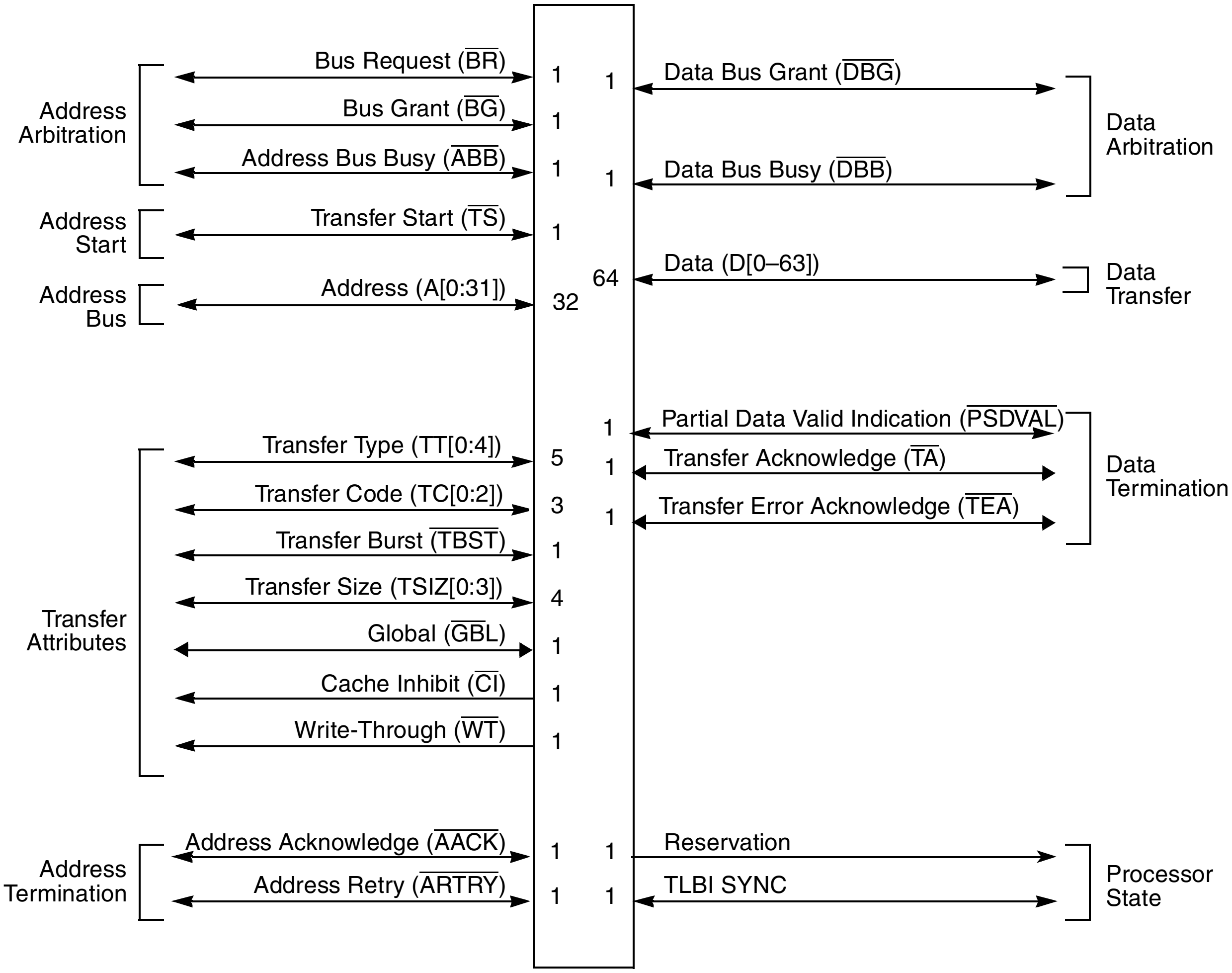


Figure 7-1. Signal Groupings

7.2 Signal Descriptions

This section describes individual MPC8272 60x signals, grouped according to Figure 7-1. Note that the following sections briefly summarize signal functions. Chapter 8, "The 60x Bus," describes many of these signals in greater detail, both in terms of their function and how groups of signals interact.

7.2.1 Address Bus Arbitration Signals

The address arbitration signals are a collection of input and output signals devices used to request address bus mastership, recognize when the request is granted, and indicate to other devices when mastership is granted. For a detailed description of how these signals interact, see Section 8.4.1, “Address Arbitration.”

Bus arbitration signals have no meaning in internal-only mode.

7.2.1.1 Bus Request ($\overline{\text{BR}}$)—Output

The bus request ($\overline{\text{BR}}$) signal is both an input and an output signal on the MPC8272.

7.2.1.1.1 Address Bus Request ($\overline{\text{BR}}$)—Output

The following are the state meaning and timing comments for the $\overline{\text{BR}}$ signal output:

State Meaning Asserted—Indicates that MPC8272 is requesting mastership of the address bus. Note that $\overline{\text{BR}}$ may be asserted for one or more cycles and then deasserted due to an internal cancellation of the bus request (for example, due to a load hit in the touch load buffer). See Section 8.4.1, “Address Arbitration.”

Negated—Indicates that the MPC8272 is not requesting the address bus. The MPC8272 may have no bus operation pending, it may be parked, or the $\overline{\text{ARTRY}}$ input was asserted on the previous bus clock cycle.

Timing Comments Assertion—May occur on any cycle; does not occur if the MPC8272 is parked and the address bus is idle ($\overline{\text{BG}}$ asserted and $\overline{\text{ABB}}$ input negated).

Negation—Occurs for at least one cycle following a qualified $\overline{\text{BG}}$ even if another transaction is pending; also negated for at least one cycle following any qualified $\overline{\text{ARTRY}}$ on the bus unless MPC8272 asserted $\overline{\text{ARTRY}}$ and requires a snoop copyback; may also be negated if MPC8272 cancels the bus request internally before receiving a qualified $\overline{\text{BG}}$.

High impedance—Occurs during a hard reset or checkstop condition

7.2.1.1.2 Address Bus Request ($\overline{\text{BR}}$)—Input

Following are the state meaning and timing comments for the $\overline{\text{BR}}$ signal input.

State Meaning Asserted—Indicates that the external master has a bus transaction to perform and is waiting for a qualified $\overline{\text{BG}}$ to begin the address tenure. $\overline{\text{BR}}$ may be asserted even if the two possible pipelined address tenures have already been granted.

Negated—Indicates that the external master has no bus transaction to perform, or if the device is parked, that it is potentially ready to start a bus transaction on the next clock cycle (with proper qualification, see $\overline{\text{BG}}$).

Timing Comments Assertion—May occur on any cycle; does not occur if the external master is parked and the address bus is idle ($\overline{\text{BG}}$ asserted and $\overline{\text{ABB}}$ input negated).

Negation—Occurs for at least one cycle after a qualified $\overline{\text{BG}}$ even if another transaction is pending; also negated for at least one cycle following any qualified $\overline{\text{ARTRY}}$ on the bus unless this chip asserted the $\overline{\text{ARTRY}}$ and requires to perform a snoop copyback; may also be negated if the external master cancels a bus request internally before receiving a qualified $\overline{\text{BG}}$.

High Impedance—Occurs during a hard reset or checkstop condition.

7.2.1.2 Bus Grant ($\overline{\text{BG}}$)

The address bus grant ($\overline{\text{BG}}$) signal is both an input and an output signal.

7.2.1.2.1 Bus Grant ($\overline{\text{BG}}$)—Input

The following are the state meaning and timing comments for the $\overline{\text{BG}}$ signal input

State Meaning Asserted—Indicates that the MPC8272 may, with the proper qualification, begin a bus transaction and assume ownership of the address bus. A qualified bus grant is generally determined from the bus state as follows: QBG = $\overline{\text{BG}}$ • ¬$\overline{\text{ABB}}$ • ¬$\overline{\text{ARTRY}}$ where $\overline{\text{ARTRY}}$ is asserted only during the cycle after $\overline{\text{AACK}}$. Note that the assertion of $\overline{\text{BR}}$ is not required for a qualified bus grant (for bus parking).

Negated—Indicates that the MPC8272 is not granted next address ownership.

Timing Comments Assertion—May occur on any cycle. Once the MPC8272 has assumed address bus ownership, it does not begin checking for $\overline{\text{BG}}$ again until the cycle after $\overline{\text{AACK}}$.

Negation—May occur whenever the MPC8272 must be prevented from using the address bus. The MPC8272 may still assume address bus ownership on the cycle $\overline{\text{BG}}$ is negated if it was asserted the previous cycle with other bus grant qualifications.

7.2.1.2.2 Bus Grant ($\overline{\text{BG}}$)—Output

The following are the state meaning and timing comments for the $\overline{\text{BG}}$ signal output:

State Meaning Asserted—Indicates that the external device may, with the proper qualification, begin a bus transaction and assume ownership of the address bus. A qualified bus grant is generally determined from the bus state as follows: QBG = $\overline{\text{BG}}$ • ¬$\overline{\text{ABB}}$ • ¬$\overline{\text{ARTRY}}$ where $\overline{\text{ARTRY}}$ is asserted only during the cycle after $\overline{\text{AACK}}$. Note that the assertion of $\overline{\text{BR}}$ is not required for a qualified bus grant (for bus parking).

Negated—Indicates that the external device is not granted next address ownership.

Timing Comments Assertion—May occur on any cycle. Once the external device has assumed address bus ownership, it does not begin checking for $\overline{\text{BG}}$ again until the cycle after $\overline{\text{AACK}}$.

Negation—May occur when an external device must be kept from using the address bus. The external device may still assume address bus ownership on the cycle that $\overline{\text{BG}}$ is negated if it was asserted the previous cycle with other bus grant qualifications.

7.2.1.3 Address Bus Busy (ABB)

The address bus busy (ABB) signal is both an input and an output signal.

7.2.1.3.1 Address Bus Busy (ABB)—Output

The following are the state meaning and timing comments for the ABB output signal:

State Meaning Asserted—Indicates that the MPC8272 is the current address bus master. The MPC8272 may not assume address bus ownership in case a bus request is internally cancelled by the cycle a qualified BG would have been recognized.

Negated—Indicates that MPC8272 is not the current address bus master.

Timing Comments Assertion—Occurs the cycle after a qualified BG is accepted by MPC8272 and remains asserted for the duration of the address tenure.

Turn-off sequencing—Negates for a fraction of a bus cycle (1/2 minimum, depends on clock mode) starting the cycle following the assertion of AACK. It then goes to the high impedance state.

7.2.1.3.2 Address Bus Busy (ABB)—Input

Following are the state meaning and timing comments for the ABB input signal.

State Meaning Asserted—Indicates that external device is the address bus master.

Negated—Indicates that the address bus may be available for use by the MPC8272 (see BG). The MPC8272 also tracks the state of ABB on the bus from the TS and AACK inputs. (See section on address arbitration phase.)

Timing Comments Assertion—May occur whenever the MPC8272 must be prevented from using the address bus.

Negation—May occur whenever the MPC8272 may use the address bus.

7.2.2 Address Transfer Start Signal

In the internal-only mode, the address transfer start signal has no meaning. Address transfer start signals are input and output signals that indicate that an address bus transfer has begun.

7.2.2.1 Transfer Start (TS)

The TS signal is both an input and an output signal on the MPC8272.

7.2.2.1.1 Transfer Start (TS)—Output

Following are the state meaning and timing comments for the TS output signal.

State Meaning Asserted—Indicates that the MPC8272 has started a bus transaction and that the address bus and transfer attribute signals are valid. It is also an implied data bus request if the transfer attributes TT[0–4] indicate that a data tenure is required for the transaction.

Negated—Has no special meaning during a normal transaction.

Timing Comments Assertion/Negation—Driven and asserted on the cycle after a qualified $\overline{\text{BG}}$ is accepted by MPC8272; remains asserted for one clock only. Negated for the remainder of the address tenure. Assertion is coincident with the first clock that $\overline{\text{ABB}}$ is asserted.

High impedance—Occurs the cycle following the assertion of $\overline{\text{AACK}}$ (same cycle as $\overline{\text{ABB}}$ negation).

7.2.2.2 Transfer Start ($\overline{\text{TS}}$)—Input

Following are the state meaning and timing comments for the $\overline{\text{TS}}$ input signal.

State Meaning Asserted—Indicates that another device has begun a bus transaction and that the address bus and transfer attribute signals are valid for snooping.

Negated—Has no special meaning.

Timing Comments Assertion/Negation—Must be asserted for one cycle only and then immediately negated. Assertion may occur at any time during the assertion of $\overline{\text{ABB}}$.

7.2.3 Address Transfer Signals

In internal only mode the memory controller uses these signals for glueless address transfers to memory and I/O devices.

The address transfer signals are used to transmit the address.

7.2.3.1 Address Bus (A[0:31])

The address bus (A[0:31]) consists of 32 signals that are both input and output signals.

7.2.3.1.1 Address Bus (A[0:31])—Output

The following are the state meaning and timing comments for the A[0:31] output signals:

State Meaning Content—Specifies the physical address of the bus transaction. For burst or extended operations, the address is a double word.

Timing Comments Assertion/Negation—Driven valid on the same cycle that $\overline{\text{TS}}$ is driven/asserted; remains driven/valid for the duration of the address tenure.

High impedance— Occurs the cycle following the assertion of $\overline{\text{AACK}}$; no precharge action performed on release.

7.2.3.1.2 Address Bus (A[0:31])—Input

Following are the state meaning and timing comments for the A[0:31] input signals.

State Meaning Asserted—Indicates that another device has begun a bus transaction and that the address bus and transfer attribute signals are valid for snooping and in slave mode.

Negated—Has no special meaning.

Timing Comments Assertion/Negation—Must be valid on the same cycle that $\overline{\text{TS}}$ is asserted; sampled by the processor only on this cycle.

7.2.4 Address Transfer Attribute Signals

In internal only mode the address transfer attribute signals have no meaning.

The transfer attribute signals are a set of signals that further characterize the transfer, such as the size of the transfer, whether it is a read or write operation, and whether it is a burst or single-beat transfer. For a detailed description of how these signals interact, see Section 7.2.4, “Address Transfer Attribute Signals.”

7.2.4.1 Transfer Type (TT[0:4])

The transfer type signals (TT[0:4]) consist of five input/output signals on the MPC8272. For a complete description of TT[0:4] signals and transfer type encoding, see Section 8.4.3.1, “Transfer Type Signal (TT[0:4]) Encoding.”

7.2.4.1.1 Transfer Type (TT[0:4])—Output

Following are the state meaning and timing comments for the TT[0:4] output signals on the MPC8272.

State Meaning Asserted/Negated—Specifies the type of transfer in progress

Timing Comments Assertion/Negation—Same as A[0:31]

High impedance—Same as A[0:31]

7.2.4.1.2 Transfer Type (TT[0:4])—Input

Following are the state meaning and timing comments for the TT[0:4] input signals on the MPC8272.

State Meaning Asserted/Negated—Specifies the type of transfer in progress for snooping by the MPC8272

Timing Comments Assertion/Negation—Same as A[0:31]

7.2.4.2 Transfer Size (TSIZ[0:3])

The transfer size (TSIZ[0:3]) signals consist of four input/output signals on the MPC8272, following are the state meaning and timing comments for the TSIZ[0:3] signals on the MPC8272.

State Meaning Asserted/Negated—Specifies the data transfer size for the transaction (see Section 8.4.3.3, “TBST and TSIZ[0:3] Signals and Size of Transfer”). During graphics transfer operations, these signals form part of the resource ID (see $\overline{\text{TBST}}$).

Timing Comments Assertion/Negation—Same as A[0:31]

High Impedance—Same as A[0:31]

7.2.4.3 Transfer Burst ($\overline{\text{TBST}}$)

The transfer burst ($\overline{\text{TBST}}$) signal is an input/output signal on the MPC8272. Following are the state meaning and timing comments for the $\overline{\text{TBST}}$ output/input signal.

State Meaning Asserted—Indicates that a burst transfer is in progress (see Section 8.4.3.3, "TBST and TSIZ[0:3] Signals and Size of Transfer"). During graphics transfer operations, this signal forms part of the resource ID field from the EAR as follows:

$\overline{\text{TBST}}$ || TSIZ[0:3] = EAR[28:31]. (See $\overline{\text{TBST}}$.)

Negated—Indicates that a burst transfer is not in progress.

Timing Comments Assertion/Negation—Same as A[0:31]

High impedance—Same as A[0:31]

7.2.4.4 Global ($\overline{\text{GBL}}$)

The global ($\overline{\text{GBL}}$) signal is an input/output signal on the MPC8272.

7.2.4.4.1 Global ($\overline{\text{GBL}}$)—Output

Following are the state meaning and timing comments for the $\overline{\text{GBL}}$ output signal.

State Meaning Asserted—Indicates that the transaction is global and should be snooped by other devices. $\overline{\text{GBL}}$ reflects the M bit (WIM bits) from the MMU except during certain transactions.

Negated—Indicates that the transaction is not global and should not be snooped by other devices.

Timing Comments Assertion/Negation—Same as A[0:31]

High impedance—Same as A[0:31]

7.2.4.4.2 Global ($\overline{\text{GBL}}$)—Input

Following are the state meaning and timing comments for the $\overline{\text{GBL}}$ input signal.

State Meaning Asserted—Indicates that a transaction must be snooped by MPC8272.

Negated—Indicates that a transaction should not be snooped by MPC8272. (In addition, certain non-global transactions are snooped for reservation coherency.)

Timing Comments Assertion/Negation—Same as A[0:31]

7.2.4.5 Caching-Inhibited ($\overline{\text{CI}}$)—Output

The cache inhibit ($\overline{\text{CI}}$) signal is an output signal on the MPC8272. Following are the state meaning and timing comments for $\overline{\text{CI}}$.

State Meaning Asserted—Indicates that the transaction in progress should not be cached. CI reflects the I bit (WIM bits) from the MMU except during certain transactions.

Negated—Indicates that the transaction should be cached.

Timing Comments Assertion/Negation—Same as A[0:31]

High impedance—Same as A[0:31]

7.2.4.6 Write-Through ($\overline{\text{WT}}$)—Output

The write-through ($\overline{\text{WT}}$) signal is an output signal on the MPC8272. Following are the state meaning and timing comments for $\overline{\text{WT}}$.

State Meaning Asserted—Indicates that the transaction should operate in write-through mode. $\overline{\text{WT}}$ reflects the W bit (WIM bits) from the MMU except during certain transactions. $\overline{\text{WT}}$ may be asserted during read transactions.

Negated—Indicates that the transaction should not operate in write-through mode.

Timing Comments Assertion/Negation—Same as A[0:31]

High impedance—Same as A[0:31]

7.2.5 Address Transfer Termination Signals

The address transfer termination signals are used to indicate either that the address phase of the transaction has completed successfully or that it must be repeated, and when it should be terminated. For detailed information about how these signals interact, see Section 7.2.5, “Address Transfer Termination Signals.”

The address transfer termination signals have no meaning in internal-only mode.

7.2.5.1 Address Acknowledge ($\overline{\text{AACK}}$)

The address acknowledge ($\overline{\text{AACK}}$) signal is an input/output on the MPC8272.

7.2.5.1.1 Address Acknowledge ($\overline{\text{AACK}}$)—Output

.Following are the state meaning and timing comments for $\overline{\text{AACK}}$ as an output signal.

State Meaning Asserted—Indicates that the address tenure of a transaction is terminated. On the cycle following the assertion of $\overline{\text{AACK}}$, the bus master releases the address-tenure-related signals to the high-impedance state and samples $\overline{\text{ARTRY}}$.

Negated—Indicates that the address bus and the transfer attributes must remain driven, if negated during $\overline{\text{ABB}}$.

Timing Comments Assertion—Occurs a programmable number of clocks after $\overline{\text{TS}}$ or whenever $\overline{\text{ARTRY}}$ conditions are resolved.

Negation—Occurs one clock after assertion.

7.2.5.1.2 Address Acknowledge ($\overline{\text{AACK}}$)—Input

Following are the state meaning and timing comments for $\overline{\text{AACK}}$ as an input signal.

State Meaning Asserted—Indicates that a 60x bus slave is terminating the address tenure. On the cycle following the assertion of $\overline{\text{AACK}}$, the bus master releases the address tenure related signals to the high-impedance state and samples $\overline{\text{ARTRY}}$.

Negated—Indicates that the address tenure must remain active and the address tenure related signals driven.

Timing Comments Assertion—Occurs during the 60x bus slave access, at least two clocks after TS.

Negation—Occurs one clock after assertion.

7.2.5.2 Address Retry (ARTRY)

The address retry (ARTRY) signal is both an input and output signal on the PowerQUICC II.

7.2.5.2.1 Address Retry (ARTRY)—Output

Following are the state meaning and timing comments for ARTRY as an output signal.

State Meaning Asserted—Indicates that the MPC8272 detects a condition in which an address tenure must be retried. If the MPC8272 processor needs to update memory as a result of snoop that caused the retry, the MPC8272 asserts BR the second cycle after AACK if ARTRY is asserted.

High impedance—Indicates that the MPC8272 does not need the address tenure to be retried.

Timing Comments Assertion—Asserted the third bus cycle following the assertion of TS if a retry is required.

Negation—Occurs the second bus cycle after the assertion of AACK. Since this signal may be simultaneously driven by multiple devices, it negates in a unique fashion. First the buffer goes to high impedance for a minimum of one-half processor cycle (dependent on the clock mode), then it is driven negated for one bus cycle before returning to high impedance.

7.2.5.2.2 Address Retry (ARTRY)—Input

Following are the state meaning and timing comments for the ARTRY input.

State Meaning Asserted—If the MPC8272 is the address bus master, ARTRY indicates that the MPC8272 must retry the preceding address tenure and immediately negate BR (if asserted). If the associated data tenure has started, the MPC8272 also aborts the data tenure immediately even if the burst data has been received. If the MPC8272 is not the address bus master, this input indicates that the MPC8272 should negate BR for one bus clock cycle immediately after the external device asserts ARTRY to permit a copy-back operation to main memory. Note that the subsequent address presented on the address bus may not be the one that generated the assertion of ARTRY.

Negated/High impedance—Indicates that the MPC8272 does not need to retry the last address tenure.

Timing Comments Assertion—May occur as early as the second cycle following the assertion of TS and must occur by the bus clock cycle immediately following the assertion of AACK if an address retry is required.

Negation—Must occur during the second cycle after the assertion of AACK.

7.2.6 Data Bus Arbitration Signals

The data bus arbitration signals have no meaning in internal-only mode.

Like the address bus arbitration signals, data bus arbitration signals maintain an orderly process for determining data bus mastership. Note that there is no data bus arbitration signal equivalent to the address bus arbitration signal $\overline{\text{BR}}$ (bus request) because, except for address-only transactions, $\overline{\text{TS}}$ implies data bus requests. For a detailed description on how these signals interact, see Section 8.5.1, "Data Bus Arbitration."

7.2.6.1 Data Bus Grant ($\overline{\text{DBG}}$)

The data bus grant signal ($\overline{\text{DBG}}$) is an output/input on the MPC8272.

7.2.6.1.1 Data Bus Grant ($\overline{\text{DBG}}$)—Input

$\overline{\text{DBG}}$ an input when MPC8272 is configured to an external arbiter. The following are the state meaning and timing comments for $\overline{\text{DBG}}$:

State Meaning Asserted—Indicates that the MPC8272 may, with the proper qualification, assume mastership of the data bus. The PowerQUICC II derives a qualified data bus grant when $\overline{\text{DBG}}$ is asserted and $\overline{\text{DBB}}$ and $\overline{\text{ARTRY}}$ are negated; that is, the data bus is not busy ($\overline{\text{DBB}}$ is negated), and there is no outstanding attempt to perform an $\overline{\text{ARTRY}}$ of the associated address tenure.

Negated—Indicates that the MPC8272 must hold off its data tenures

Timing Comments Assertion—May occur any time to indicate the MPC8272 is free to take data bus mastership. It is not sampled until $\overline{\text{TS}}$ is asserted.

Negation—May occur at any time to indicate the MPC8272 cannot assume data bus mastership

7.2.6.1.2 Data Bus Grant ($\overline{\text{DBG}}$)—Output

$\overline{\text{DBG}}$ signal is output when the MPC8272 configured to use the internal arbiter. Following are the state meaning and timing comments for the $\overline{\text{DBG}}$ signal.

State Meaning Asserted—Indicates that the external device may, with the proper qualification, assume mastership of the data bus. A qualified data bus grant is defined as the assertion of $\overline{\text{DBG}}$, negation of $\overline{\text{DBB}}$, and negation of $\overline{\text{ARTRY}}$. The requirement for the $\overline{\text{ARTRY}}$ signal is only for the address bus tenure associated with the data bus tenure about to be granted (that is, not for another address tenure available because of address pipelining).

Negated—Indicates that an external device is not granted mastership of the data bus

Timing Comments Assertion—Occurs on the first clock in which the data bus is not busy and the processor has the highest priority outstanding data transaction

Negation—Occurs one clock after assertion

7.2.6.2 Data Bus Busy ($\overline{\text{DBB}}$)

The data bus busy ($\overline{\text{DBB}}$) signal is both an input and output signal on the MPC8272.

7.2.6.2.1 Data Bus Busy ($\overline{\text{DBB}}$)—Output

Following are the state meaning and timing comments for the $\overline{\text{DBB}}$ output signal.

State Meaning Asserted—Indicates that the MPC8272 is the data bus master. The MPC8272 always assumes data bus mastership if it needs the data bus and determines a qualified data bus grant (see $\overline{\text{DBG}}$).

Negated—Indicates that the MPC8272 is not using the data bus

Timing Comments Assertion—Occurs during the bus clock cycle following a qualified $\overline{\text{DBG}}$

Negation—Occurs for a minimum of one-half bus clock cycle following the assertion of the final $\overline{\text{TA}}$ following $\overline{\text{TEA}}$ or certain $\overline{\text{ARTRY}}$ cases

High impedance—Occurs after $\overline{\text{DBB}}$ is negated

7.2.6.2.2 Data Bus Busy ($\overline{\text{DBB}}$)—Input

Following are the state meaning and timing comments for the $\overline{\text{DBB}}$ input signal.

State Meaning Asserted—Indicates that another device is bus master.

Negated—Indicates that the data bus is free (with proper qualification, see $\overline{\text{DBG}}$) for use by the MPC8272.

Timing Comments Assertion—Must occur when the MPC8272 must be prevented from using the data bus.

Negation—May occur whenever the data bus is available.

7.2.7 Data Transfer Signals

Data transfer signals are used in the same way in both internal-only and external-master modes. Like the address transfer signals, the data transfer signals are used to transmit data and to generate and monitor parity for the data transfer. For a detailed description of how data transfer signals interact, see Section 7.2.7, “Data Transfer Signals.”

7.2.7.1 Data Bus (D[0:63])

The data bus (D[0:63]) states have the same meanings in both internal only mode and external master mode. The data bus consists of 64 signals that are both inputs and outputs on the MPC8272. Following are the state meaning and timing comments for the data bus.

State Meaning The data bus holds 8 byte lanes assigned as shown in Table 7-1.

Timing Comments The number of times the data bus is driven depends on the transfer size, port size, and whether the transfer is a single-beat or burst operation.

7.2.7.1.1 Data Bus (D[0:63])—Output

Following are the state meaning and timing comments for the D[0:63] output signals.

State Meaning Asserted/Negated—Represents the state of data during a data write. Byte lanes not selected for data transfer do not supply valid data. MPC8272 duplicates data to enable valid data to be sent to different port sizes.

Timing Comments Assertion/Negation—Initial beat coincides with $\overline{\text{DBB}}$, for bursts, transitions on the bus clock cycle following each assertion of $\overline{\text{TA}}$ and, for port size, transitions on the bus clock cycle following each assertion of $\overline{\text{PSDVAL}}$.

High impedance—Occurs on the bus clock cycle after the final assertion of $\overline{\text{TA}}$, $\overline{\text{TEA}}$, or certain $\overline{\text{ARTRY}}$ cases.

Table 7-1. Data Bus Lane Assignments

Data Bus Signals	Byte Lane
D0:D7	0
D8:D15	1
D16:D23	2
D24:D31	3
D32:D39	4
D40:D47	5
D48:D55	6
D56:D63	7

7.2.7.1.2 Data Bus (D[0:63])—Input

Following are the state meaning and timing comments for the D[0:63] input signals.

State Meaning Asserted/Negated—Represents the state of data during a data read transaction

Timing Comments Assertion/Negation—Data must be valid on the same bus clock cycle that $\overline{\text{TA}}$ and/or $\overline{\text{PSDVAL}}$ is asserted.

7.2.8 Data Transfer Termination Signals

Data termination signals are required after each data beat in a data transfer. Note that in a single-beat transaction that is not a port-size transfer, the data termination signals also indicate the end of the tenure. In burst or port size accesses, the data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat. For a detailed description of how these signals interact, see Section 8.5, “Data Tenure Operations.”

7.2.8.1 Transfer Acknowledge ($\overline{\text{TA}}$)

The transfer acknowledge ($\overline{\text{TA}}$) signal is both input and output on the MPC8272.

7.2.8.1.1 Transfer Acknowledge ($\overline{\text{TA}}$)—Input

Following are the state meaning and timing comments for the $\overline{\text{TA}}$ input signal.

State Meaning Asserted—Indicates that a single-beat data transfer completed successfully or that a data beat in a burst transfer completed successfully. Note that $\overline{\text{TA}}$ must be asserted for each data beat in a burst transaction. For more information, see Section 8.5.3, "Data Bus Transfers and Normal Termination."

Negated—(During assertion of $\overline{\text{DBB}}$) indicates that, until $\overline{\text{TA}}$ is asserted, the MPC8272 must continue to drive the data for the current write or must wait to sample the data for reads.

Timing Comments Assertion—If the address retry mechanism is to be used to prevent invalid data from being used by the MPC8272 and the PCI controller can initiate global transactions, assertion must occur at least one clock cycle following $\overline{\text{AACK}}$ for the current transaction and at least one clock cycle after $\overline{\text{ARTRY}}$ can be asserted.

Negation—Must occur after the bus clock cycle of the final (or only) data beat of the transfer. For a burst transfer, the system can assert $\overline{\text{TA}}$ for one bus clock cycle and then negate it to advance the burst transfer to the next beat and insert wait states during the next beat. (Note: when configured for 1:1 clock mode and is performing a burst read into the data cache, the MPC8272 requires two wait states between the assertion of $\overline{\text{TS}}$ and the first assertion of $\overline{\text{TA}}$ for that transaction, or one wait state for 1.5:1 clock mode.)

7.2.8.1.2 Transfer Acknowledge ($\overline{\text{TA}}$)—Output

Following are the state meaning and timing comments for $\overline{\text{TA}}$ as an output signal.

State Meaning Asserted—Indicates that the data has been latched for a write operation, or that the data is valid for a read operation, thus terminating the current data beat. If it is the last or only data beat, this also terminates the data tenure.

Negated—Indicates that master must extend the current data beat (insert wait states) until data can be provided or accepted by the MPC8272.

Timing Comments Assertion—If the address retry mechanism is to be used to prevent invalid data from being used by the MPC8272 and the PCI controller can initiate global transactions, assertion must occur at least one clock cycle following $\overline{\text{AACK}}$ for the current transaction and at least one clock cycle after $\overline{\text{ARTRY}}$ can be asserted.

Negation—Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, $\overline{\text{TA}}$ may be negated between beats to insert one or more wait states before the completion of the next beat.

7.2.8.2 Transfer Error Acknowledge ($\overline{\text{TEA}}$)

The transfer error acknowledge ($\overline{\text{TEA}}$) signal is both input and output on the PowerQUICC II.

7.2.8.2.1 Transfer Error Acknowledge (TEA)—Input

Following are the state meaning and timing comments for the TEA input signal.

State Meaning Asserted—Indicates that a bus error occurred. The assertion of TEA causes the negation/high impedance of DBB in the next clock cycle. However, data entering the MPC8272 internal memory resources such as GPRs or caches are not invalidated.

Negated—Indicates that no bus error was detected

Timing Comments Assertion—May be asserted while DBB is asserted and for the cycle after is TA is asserted during a read operation. TEA should be asserted for one cycle only.

Negation—TEA must be negated no later than the negation of DBB.

7.2.8.2.2 Transfer Error Acknowledge (TEA)—Output

Following are the state meaning and timing comments for the TEA output.

State Meaning Asserted—Indicates that a bus error has occurred. Assertion of TEA terminates the transaction in progress; that is, asserting TA is unnecessary because it is ignored by the target device. An unsupported memory transaction, such as a direct-store access or a graphics read or write, causes the assertion of TEA (provided TEA is enabled and the address transfer matches the MPC8272 memory map).

Negated—Indicates that no bus error was detected

Timing Comments Assertion—Occurs on the first clock after the bus error is detected

Negation—Occurs one clock after assertion

7.2.8.3 Partial Data Valid Indication (PSDVAL)

The partial data valid indication (PSDVAL) is both an input and output on the MPC8272.

7.2.8.3.1 Partial Data Valid (PSDVAL)—Input

Following are the state meaning and timing comments for the PSDVAL input signal. Note that TA asserts with PSDVAL to indicate the termination of the current transfer and for each complete data beat in burst transactions.

State Meaning Asserted—Indicates that a beat data transfer completed successfully. Note that PSDVAL must be asserted for each data beat in a single beat, port size and burst transaction,. For more information, see Section 8.5.5, "Port Size Data Bus Transfers and PSDVAL Termination."

Negated—(During DBB) indicates that, until PSDVAL is asserted, the MPC8272 must continue to drive the data for the current write or must wait to sample the data for reads.

Timing Comments Assertion—Must not occur before AACK for the current transaction (if the address retry mechanism is to be used to prevent invalid data from being used by the MPC8272); otherwise, assertion may occur at any time during the assertion of

$\overline{\text{DBB}}$. The system can withhold assertion of $\overline{\text{PSDVAL}}$ to indicate that the MPC8272 should insert wait states to extend the duration of the data beat.

Negation—Must occur after the bus clock cycle of the final (or only) data beat of the transfer. For a burst and/or port size transfer, the system can assert $\overline{\text{PSDVAL}}$ for one bus clock cycle and then negate it to insert wait states during the next beat. (Note: when the MPC8272 processor is configured for 1:1 clock mode and is performing a burst read into the data cache, the MPC8272 requires two wait state between the assertion of $\overline{\text{TS}}$ and the first assertion of $\overline{\text{PSDVAL}}$ for that transaction, or one wait state for 1.5:1 clock mode.)

7.2.8.3.2 Partial Data Valid ($\overline{\text{PSDVAL}}$)—Output

Following are the state meaning and timing comments for $\overline{\text{PSDVAL}}$ as an output signal.

State Meaning Asserted—Indicates that the data has been latched for a write operation, or that the data is valid for a read operation, thus terminating the current data beat. If it is the last or only data beat, this also terminates the data tenure.

Negated—Indicates that the master must extend the current data beat (insert wait states) until data can be provided or accepted by the MPC8272.

Timing Comments Assertion—Occurs on the clock in which the current data transfer can be completed.

Negation—Occurs after the clock cycle of the final (or only) data beat of the transfer. For a burst transfer, $\overline{\text{PSDVAL}}$ may be negated between beats to insert one or more wait states before the completion of the next beat.

Chapter 8
The 60x Bus

The 60x bus, which is used by processors that implement the PowerPC architecture, provides flexible support for the on-chip G2_LE processor, as well as other internal and external bus devices. The 60x bus supports 32-bit addressing, a 64-bit data bus, and burst operations that transfer as many as 256 bits of data in a four-beat burst. The 60x data bus can be accessed in 8-, 16-, 32-, and 64-bit data ports. The 60x bus supports accesses of 1, 2, 3, and 4 bytes, aligned or unaligned, on 4-byte (word) boundaries; it also supports 64-, 128-, 192-, and 256-bit accesses.

The address and data buses support synchronous, one-level pipeline transactions. The 60x bus interface can be configured to support both external and internal masters or internal masters only.

8.1 Terminology

Table 8-1 defines terms used in this chapter.

Table 8-1. Terminology

Term	Definition
Atomic	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 MPC8272 initiates the read and write separately, but signals the memory system that it is attempting an atomic operation. If the operation fails, status is kept so that MPC8272 can try again.
Beat	A single state on the MPC8272 interface that may extend across multiple bus cycles. (An MPC8272 transaction can be composed of multiple address or data beats.)
Burst	A multiple-beat data transfer whose total size is typically equal to a cache block size (in the MPC8272: 32 bytes, or 4 data beats at 8 bytes per beat).
Cache block	The PowerPC architecture defines the basic unit of coherency as a cache block, which can be considered the same thing as a cache line.
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.
Flush	An operation that causes a cache block to be invalidated in the cache, and its data, if modified, to be written back to main memory.
Kill	An operation that causes a cache block to be invalidated in the cache without writing any modified data to memory.
Lane	A sub-grouping of signals within a bus. An 8-bit section of the address or data bus may be referred to as a byte lane for that bus.
Master	The device that owns the address or data bus, the device that initiates or requests the transaction.
Modified	Identifies a cache block The M state in a MESI or MEI protocol.

Table 8-1. Terminology (continued)

Term	Definition
Parking	Granting potential bus mastership without requiring a bus request from that device. This eliminates the arbitration delay associated with the bus request.
Pipelining	Initiating a bus transaction before the current one finishes. This involves running an address tenure for a new bus transaction before the data tenure for a current bus transaction completes.
Slave	The device addressed by the master. The slave is identified in the address tenure and is responsible for sourcing or sinking 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.
Split-transaction	A transaction with separate request and response tenures.
Tenure	The period of bus mastership. For the MPC8272, there can be separate address bus tenures and data bus tenures.
Transaction	A complete exchange between two bus devices. A typical transaction is composed of an address tenure and a data tenure, which may overlap or occur separately from the address tenure. A transaction can minimally consist of an address tenure alone.

8.2 Bus Configuration

The 60x bus supports separate bus configurations for internal masters and external bus masters.

- Single-MPC8272 bus mode connects external devices by using only the memory controller. This is described in Section 8.2.1, “Single-MPC8272 Bus Mode.”
- The 60x-compatible bus mode, described in Section 8.2.2, “60x-Compatible Bus Mode,” enables connections to other masters and 60x-bus slaves.

The figures in the following sections show how the MPC8272 can be connected in these two configurations.

8.2.1 Single-MPC8272 Bus Mode

In single-MPC8272 bus mode, the MPC8272 is the only bus device in the system. The internal memory controller controls all devices on the external pins. Figure 8-1 shows the signal connections for single-MPC8272 bus mode.

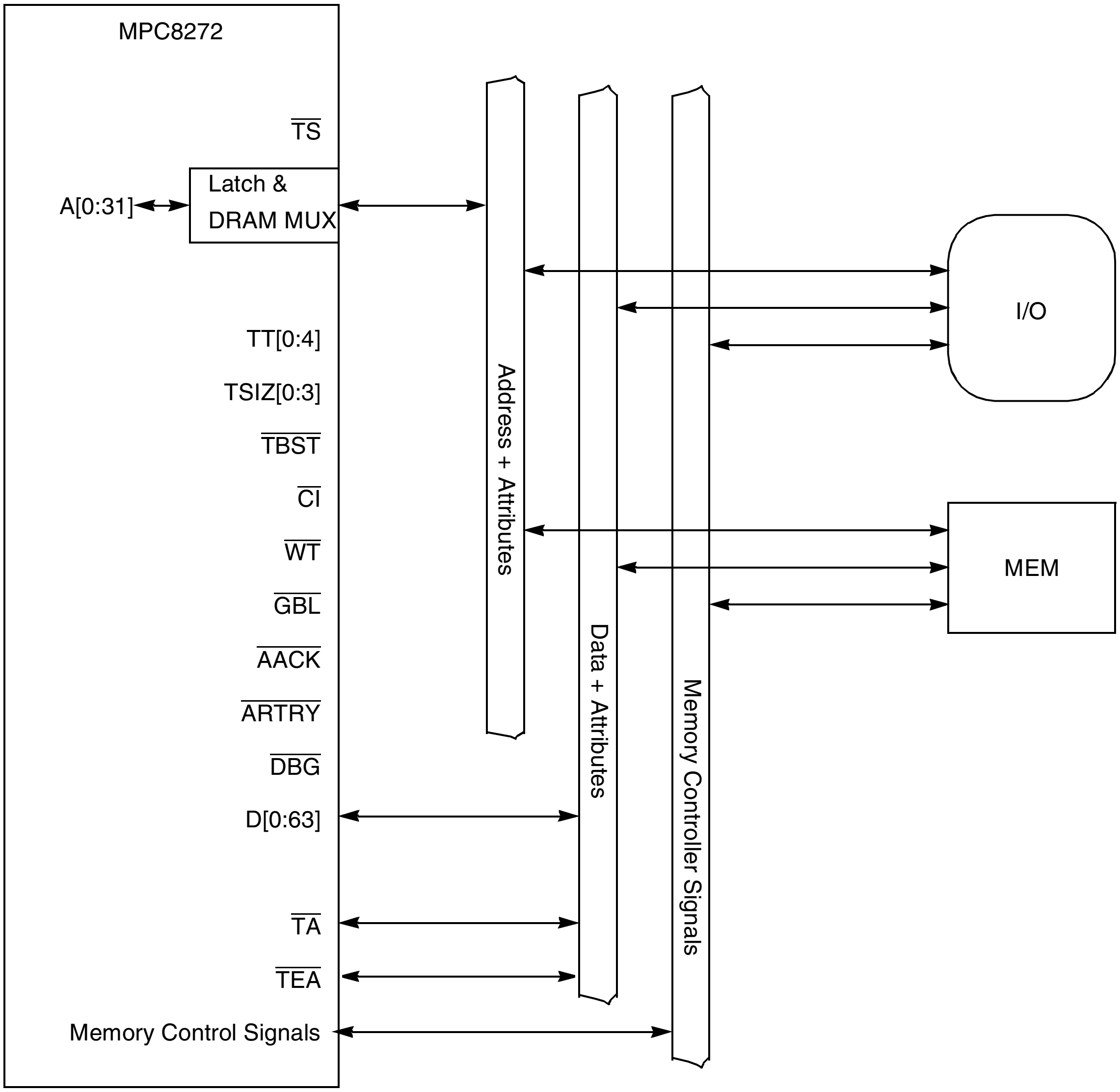


Figure 8-1. Single-MPC8272 Bus Mode

NOTE

In single-MPC8272 bus mode, the MPC8272 uses the address bus as a memory address bus. Slaves cannot use the 60x bus signals because the addresses have memory timing, not address tenure timing.

8.2.2 60x-Compatible Bus Mode

The 60x-compatible bus mode can include one or more potential external masters (for example, an ASIC DMA). Note that the external L2 cache is not supported in this device. When operating in a multiprocessor configuration, the MPC8272 snoops bus operations and maintains coherency between the primary caches and main memory. Figure 8-2 shows how an external processor is attached to the MPC8272.

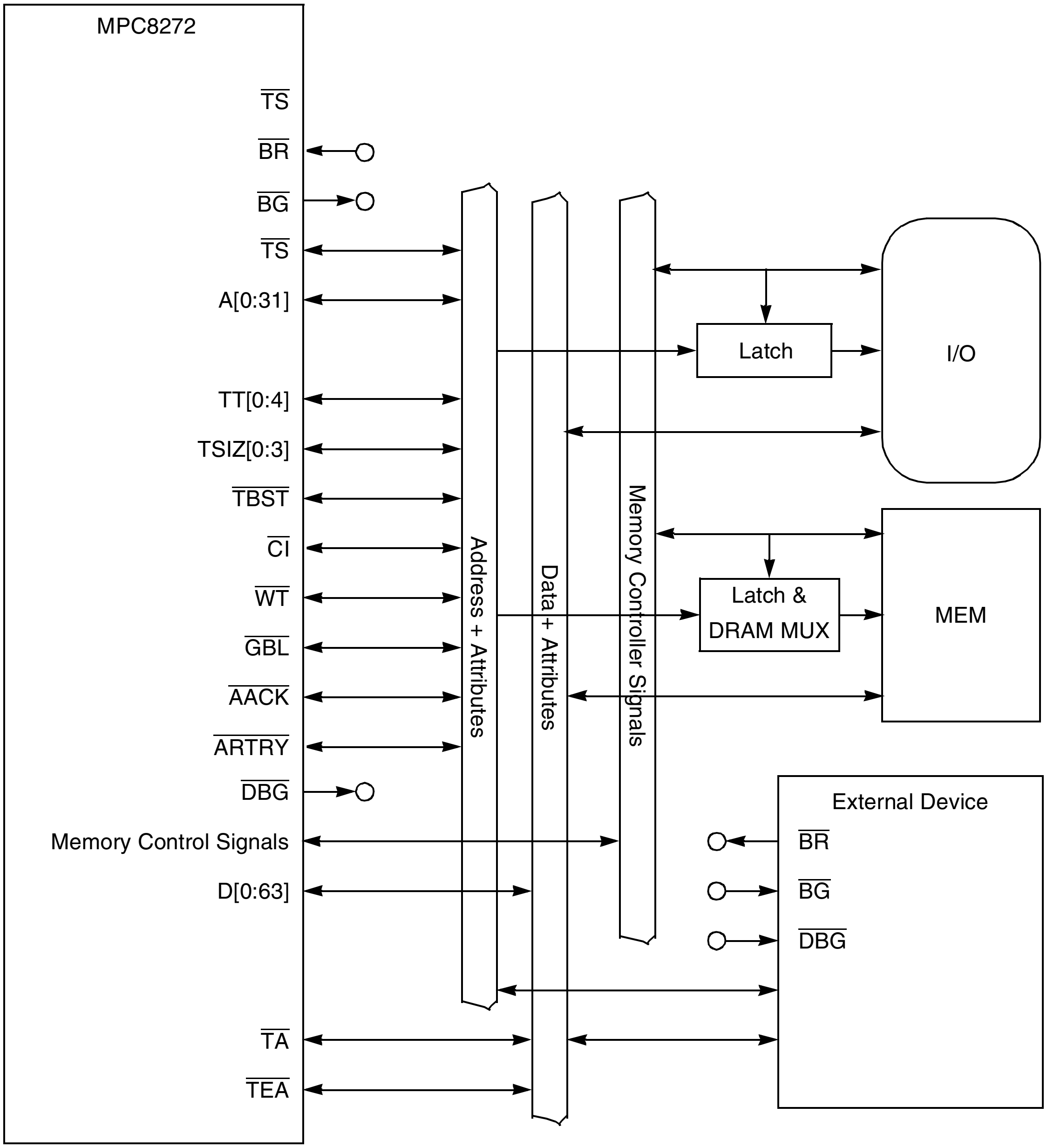


Figure 8-2. 60x-Compatible Bus Mode

8.3 60x Bus Protocol Overview

Typically, 60x bus accesses consist of address and data tenures, which in turn each consist of three phases—arbitration, transfer, and termination, as shown in Figure 8-3. The independence of the tenures is indicated by showing the data tenure overlap the next address tenure, which allows split-bus transactions to be implemented at the system level in multiprocessor systems. Figure 8-3 shows a data transfer that consists of a single-beat transfer of as many as 256 bits. Four-beat burst transfers of 32-byte cache blocks require data transfer termination signals for each beat of data. Note that the MPC8272 supports port sizes of 8, 16, 32, and 64 bits and requires an additional bus signal, PSDVAL, which is not defined by the 60x bus specification. The beginning of an address transfer is marked by the assertion of the transfer start ($\overline{TS}$) signal. For more information, see Section 8.5.5, "Port Size Data Bus Transfers and PSDVAL Termination."

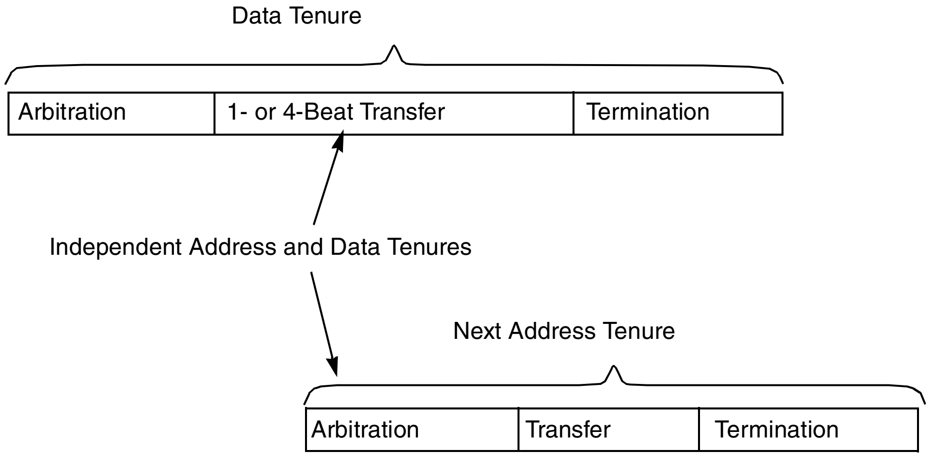


Figure 8-3. Basic Transfer Protocol

The basic functions of the address and data tenures are as follows:

- Address tenure
  - — Arbitration—Address bus arbitration signals are used to request and grant address bus mastership.
  - — Transfer—After a device is granted address bus mastership, it transfers the address. The address signals and transfer attribute signals control the address transfer.
  - — Termination—After the address transfer, the system acknowledges that the address tenure is complete or that it must be repeated, signaled by the assertion of the address retry signal ($\overline{ARTRY}$).
- Data tenure
  - — Arbitration—After the address tenure begins, the bus device arbitrates for data bus mastership.
  - — Transfer—After the device is granted data bus mastership, it samples the data bus for read operations or drives the data bus for write operations.
  - — Termination—Acknowledgment of a successful data transfer is required after each beat in a data transfer. In single-beat transactions, the data termination signals also indicate the end of the tenure. In burst or port-size accesses, data termination signals indicate the completion of individual beats and, after the final data beat, the end of the tenure.

8.3.1 Arbitration Phase

The external bus design permits one device (either the MPC8272 or a bus-attached external device) to be granted bus mastership at a time. Bus arbitration can be handled either by an external central bus arbiter or by the internal on-chip arbiter. In the latter case, the system is optimized for three external bus masters besides the MPC8272. The arbitration configuration (external or internal) is determined at system reset by sampling configuration pins. See Section 4.3.2.2, “60x Bus Arbiter Configuration Register (PPC_ACR),” for more information.

The MPC8272 controls bus access through the bus request ($\overline{\text{BR}}$) and bus grant ($\overline{\text{BG}}$) signals. It determines the state of the address and data bus busy signals by monitoring $\overline{\text{DBG}}$, $\overline{\text{TS}}$, $\overline{\text{AACK}}$, and $\overline{\text{TA}}$, and qualifies them with $\overline{\text{ABB}}$ and $\overline{\text{DBB}}$.

The following signals are used for address bus arbitration:

- $\overline{\text{BR}}$ (bus request)—A device asserts $\overline{\text{BR}}$ to request address bus mastership.
- $\overline{\text{BG}}$ (bus grant)—Assertion indicates that a bus device may, with proper qualification, assume mastership of the address bus. A qualified bus grant occurs when $\overline{\text{BG}}$ is asserted while $\overline{\text{ABB}}$ and $\overline{\text{ARTRY}}$ are negated.
- $\overline{\text{ABB}}$ (address bus busy)—A device asserts $\overline{\text{ABB}}$ to indicate it is the current address bus master. Note that if all devices assert $\overline{\text{ABB}}$ with $\overline{\text{TS}}$ and would normally negate $\overline{\text{ABB}}$ after $\overline{\text{AACK}}$ is asserted, the devices can ignore $\overline{\text{ABB}}$ because the MPC8272 can internally generate $\overline{\text{ABB}}$. The MPC8272’s $\overline{\text{ABB}}$, if enabled, must be tied to a pull-up resistor.

The following signals are used for data bus arbitration:

- $\overline{\text{DBG}}$ (data bus grant)—Indicates that a bus device can, with the proper qualification, assume data bus mastership. A qualified data bus grant occurs when $\overline{\text{DBG}}$ is asserted while $\overline{\text{DBB}}$ and $\overline{\text{ARTRY}}$ are negated.
- $\overline{\text{DBB}}$ (data bus busy)—Assertion by the device indicates that the device is the current data bus master. The device master always assumes data bus mastership if it needs the data bus and is given a qualified data bus grant (see $\overline{\text{DBG}}$). Note that if all devices assert $\overline{\text{DBB}}$ in conjunction with qualified data bus grant and would normally negate $\overline{\text{DBB}}$ after the last $\overline{\text{TA}}$ is asserted, the devices can ignore $\overline{\text{DBB}}$ because the MPC8272 can generate $\overline{\text{DBB}}$ internally. The MPC8272’s $\overline{\text{DBB}}$ signal, if enabled, must be tied to a pull-up resistor.

The following is a summary of rules for arbitration:

- Preference among devices is determined at the request level. The MPC8272 supports eight levels of bus requests.
- When no bus device is requesting the address bus, the MPC8272 parks the device selected in the arbiter configuration register on the bus.

For more information, see Section 4.3.2.2, “60x Bus Arbiter Configuration Register (PPC_ACR).”

8.3.2 Address Pipelining and Split-Bus Transactions

The 60x bus protocol provides independent address and data bus capability to support pipelined and split-bus transaction system organizations. Address pipelining allows the next address tenure to begin before the current data tenure has finished. Although this ability does not inherently reduce memory latency, support for address pipelining and split-bus transactions can greatly improve effective bus/memory throughput. These benefits are most fully realized in shared-memory, multiple-master implementations where bus bandwidth is critical to system performance.

External arbitration (as provided by the MPC8272) is required in systems in which multiple devices share the system bus. The MPC8272 uses the address acknowledge (AACK) signal to control pipelining. The MPC8272 supports both one- and zero-level bus pipelining. One-level pipelining is achieved by asserting AACK to the current address bus master and granting mastership of the address bus to the next requesting master before the current data bus tenure has completed. Two address tenures can occur before the current data bus tenure completes. The MPC8272 also supports non-pipelined accesses.

8.4 Address Tenure Operations

This section describes the three phases of the address tenure—address bus arbitration, address transfer, and address termination.

8.4.1 Address Arbitration

Bus arbitration can be handled either by an external arbiter or by the internal on-chip arbiter. The arbitration configuration (external or internal) is chosen at system reset. For internal arbitration, the MPC8272 provides arbitration for the 60x address bus, and the system is optimized for three external bus masters besides the MPC8272. The bus request (BR) for the external device is an external input to the arbiter. The bus grant signal (BG) for the external device is output to the external device. The BG signal asserted by MPC8272's on-chip arbiter is asserted 1 clock after the current master on the bus has asserted AACK; therefore, it can be called a qualified BG. Assuming that all potential masters negate ABB 1 clock after receiving AACK, the device receiving BG can start the address tenure (by asserting TS) 1 clock after receiving BG. In addition to the external signals, internal request and grant signals exist for the MPC8272 processor, communications processor, refresh controller, and the PCI internal bridge. Bus accesses are prioritized, with programmable priority. When a MPC8272's internal master needs the 60x bus, it asserts the internal bus request along with the request level. The arbiter asserts the internal bus grant for the highest priority request.

The MPC8272 supports address bus parking through the use of the parked master bits in the arbiter configuration register. The MPC8272 parks the address bus (asserts the address bus grant signal in anticipation of an address bus request) to the external master or internal masters. When a device is parked, the arbiter can hold BG asserted for a device even if that device has not requested the bus. Therefore, when the parked device needs to perform a bus transaction, it skips the bus request delay and assumes address bus mastership on the next cycle. For this case, BR is not asserted and the access latency seen by the device is shortened by one cycle.

The MPC8272 and external device bus devices qualify BG by sampling ARTRY in the negated state prior to taking address bus mastership. The negation of ARTRY during the address retry window (one cycle after the assertion of AACK) indicates that no address retry is requested. If a device detects that ARTRY is asserted, it cannot accept a address bus grant during the ARTRY cycle or the following cycle. A device that asserts ARTRY due to a modified cache block hit, for example, asserts its bus request during the cycle after the assertion of ARTRY and assumes bus mastership for the cache block push when it is given a bus grant.

The series of address transfers in Figure 8-4 shows the transfer protocol when the MPC8272 is configured in 60x-compatible bus mode. In this example, MPC8272 is initially parked on the bus with BG INT asserted (note that BG INT is an internal signal not seen by the user at the pins), which lets it start an address bus tenure by asserting $\overline{\text{TS}}$. During the same clock cycle, the external master's bus request is asserted to request access to the 60x bus, thereby causing the negation of $\overline{\text{BG INT}}$ internally and the assertion of $\overline{\text{BG}}$ at the pin. Following MPC8272's address tenure, the external master takes the bus and initiates its address transaction. The on-chip arbiter samples $\overline{\text{BR}}$ during the clock cycle in which $\overline{\text{AACK}}$ is asserted; if $\overline{\text{BR}}$ is not asserted (no pending request), it negates $\overline{\text{BG}}$ and asserts the parked bus grant ($\overline{\text{BG\_INT}}$ in this example).

The master can assert $\overline{\text{BR}}$ and receive a qualified bus grant without subsequently using the bus. It can negate (cancel) $\overline{\text{BR}}$ before accepting a qualified bus grant. This can occur when a replacement copyback transaction waiting to be run on the bus is killed by a snoop of another bus master. This can also occur when the reservation set by a pending **stwcx.** transaction is cancelled by a snoop of another master. In both cases, the pending transaction by the processor is cancelled and $\overline{\text{BR}}$ is negated.

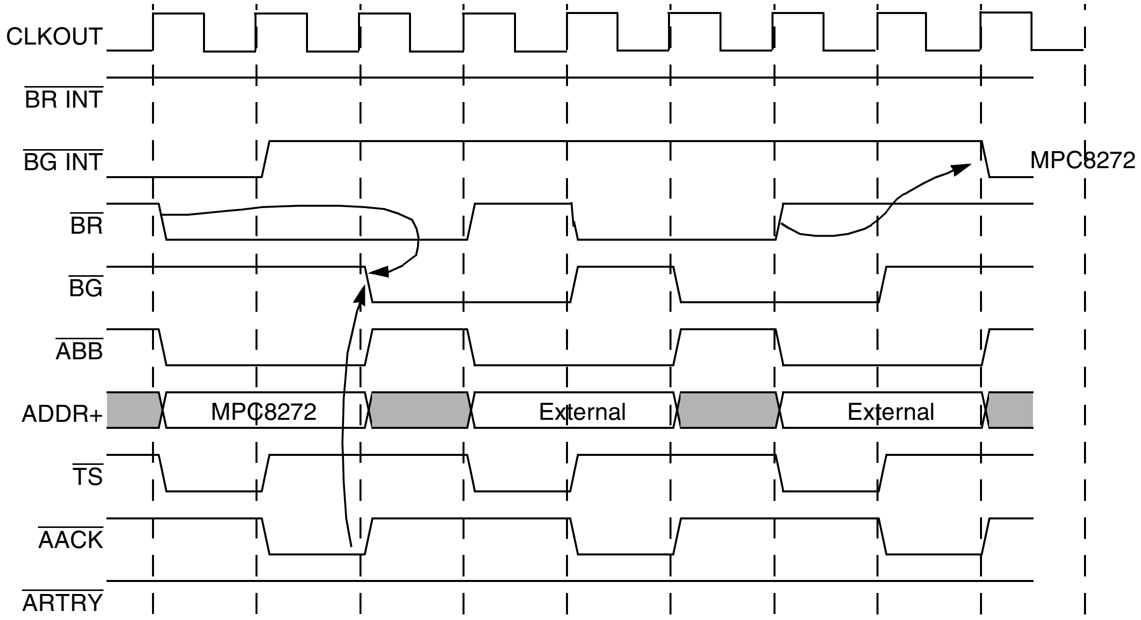


Figure 8-4. Address Bus Arbitration with External Bus Master

8.4.2 Address Pipelining

The MPC8272 supports one-level address pipelining by asserting $\overline{\text{AACK}}$ to the current bus master when its data tenure starts and by granting the address bus to the next requesting device before the current data bus tenure completes. Address pipelining improves data throughput by allowing the memory-control hardware to decode a new set of address and control signals while the current data transaction finishes. The MPC8272 pipelines data bus operations in strict order with the associated address operations. Figure 8-5 shows how address pipelining allows address tenures to overlap the associated data tenures.

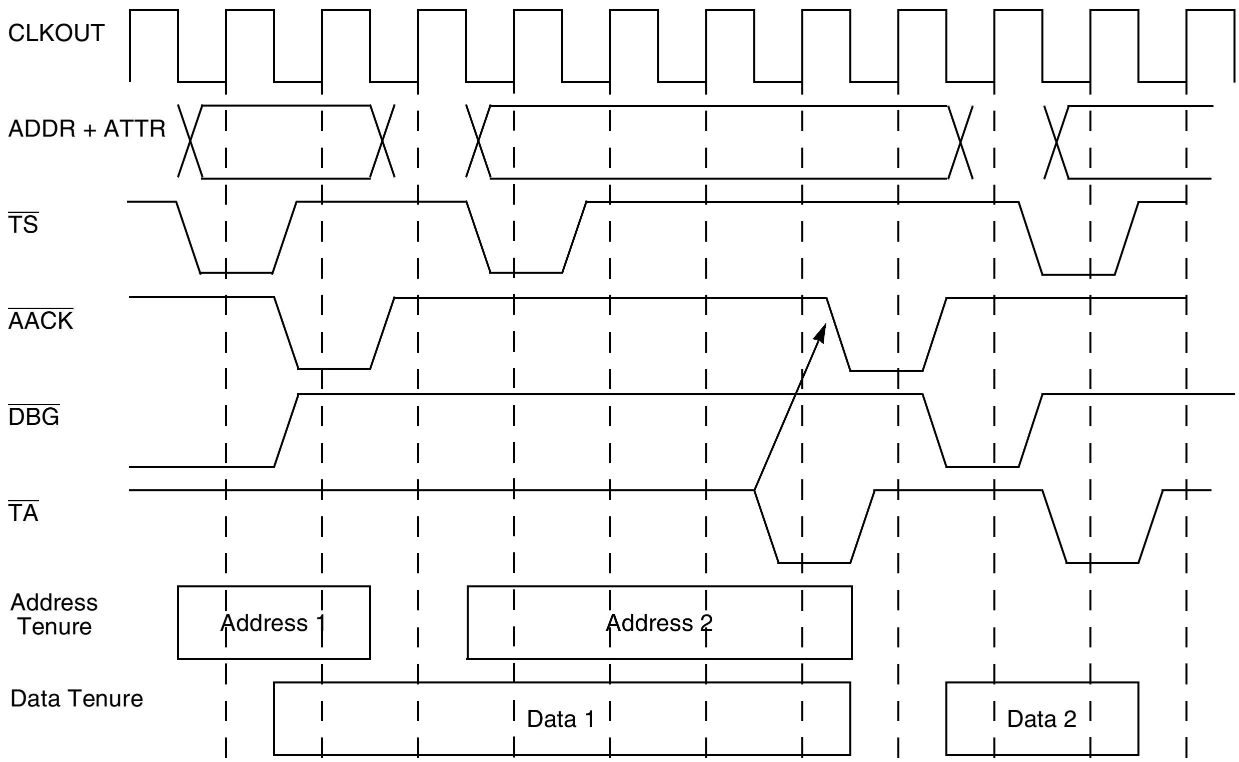


Figure 8-5. Address Pipelining

8.4.3 Address Transfer Attribute Signals

During the address transfer, the address is placed on the address signals, A[0:31]. The bus master provides other signals that characterize the address transfer—transfer type (TT[0:4]), transfer code (TC[0:2]), transfer size (TSIZ[0:3]), and transfer burst (TBST) signals. These signals are discussed in the following sections.

8.4.3.1 Transfer Type Signal (TT[0:4]) Encoding

The transfer type signals define the nature of the transfer requested. They indicate whether the operation is an address-only transaction or both address and data are to be transferred. Table 8-2 describes the MPC8272's action as master, slave, and snooper.

Table 8-2. Transfer Type Encoding

TT[0:4][1]	60x Bus Specification[2]		MPC8272 as Bus Master		MPC8272 as Snooper	MPC8272 as Slave
	Command	Transaction	Bus Trans.	Transaction Source	Action on Hit	Action on Slave Hit
00000	Clean block	Address only	Address only (if enabled)	**dcbst** (if enabled)	Not applicable	AACK asserted; MPC8272 takes no further action.
00100	Flush block	Address only	Address only (if enabled)	**dcbf** (if enabled)	Not applicable	AACK is asserted; MPC8272 takes no further action.

Table 8-2. Transfer Type Encoding (continued)

TT[0:4][1]	60x Bus Specification[2]		MPC8272 as Bus Master		MPC8272 as Snooper	MPC8272 as Slave
	Command	Transaction	Bus Trans.	Transaction Source	Action on Hit	Action on Slave Hit
01000	**sync**	Address only	Address only (if enabled)	**sync** (if enabled)	Not applicable	Assert $\overline{\text{AACK}}$. $\overline{\text{BG}}$ is negated until MPC8272 buffers are flushed.
01100	Kill block	Address only	Address only	**dcbz** or **dcbi** (if enabled)	Flush, cancel reservation	$\overline{\text{AACK}}$ is asserted.
10000	**eieio**	Address only	Address only (if enabled)	**eieio** (if enabled)	Not applicable	Assert $\overline{\text{AACK}}$. $\overline{\text{BG}}$ is negated until MPC8272 buffers are flushed.
101 00	Graphics write	Single-beat write	Single-beat write (non-GLB)	**ecowx**	Not applicable	No action
11000	TLB invalidate	Address only	Not applicable	Not applicable	Not applicable	$\overline{\text{AACK}}$ is asserted; MPC8272 takes no further action.
11100	Graphics read	Single-beat read	Single-beat read (non-GBL)	**eciwx**	Not applicable	MPC8272 takes no action.
00001	**lwarx** reservation set	Address only	Not applicable	Not applicable	Not applicable	Address-only operation. $\overline{\text{AACK}}$ is asserted; MPC8272 takes no further action.
00101	Reserved	—	Not applicable	Not applicable	Not applicable	Illegal
01001	**tlbsync**	Address only	Not applicable	Not applicable	Not applicable	Address-only operation. $\overline{\text{AACK}}$ is asserted; MPC8272 takes no further action.
01101	**icbi**	Address only	Not applicable	Not applicable	Not applicable	Address-only operation. $\overline{\text{AACK}}$ is asserted; MPC8272 takes no further action.
1XX01	Reserved for customer	—	Not applicable	Not applicable	Not applicable	Illegal
00010	WR w/ flush	Single-beat write or Burst	Single-beat write	CI, WT store, or non-processor master under	Flush, cancel reservation	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
00110	WR w/ kill	Burst	Burst (non-GLB)	Castout, ca-op push, or snoop copyback	Kill, cancel reservation	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
01010	Read	Single-beat read or burst	Single-beat read	CI load, CI I-fetch or nonprocessor master	Clean or flush	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.

Table 8-2. Transfer Type Encoding (continued)

TT[0:4][1]	60x Bus Specification[2]		MPC8272 as Bus Master		MPC8272 as Snooper	MPC8272 as Slave
	Command	Transaction	Bus Trans.	Transaction Source	Action on Hit	Action on Slave Hit
01110	Read with intent to modify	Burst	Burst	Load miss, store miss, or I-fetch	Flush	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$.
10010	WR w/ flush atomic	Single-beat write	Single-beat write	**stwcx**	Flush, cancel reservation	Write, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
10110	Reserved	Not applicable	Not applicable	Not applicable	Not applicable	Illegal
11010	Read atomic	Single-beat read or burst	Single-beat read	**lwarx** (CI load)	Clean or flush	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
11110	Read with intent to modify atomic	Burst	Burst	**lwarx** (load miss)	Flush	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
00011	Reserved	—	Not applicable	Not applicable	Not applicable	Illegal
00111	Reserved	—	Not applicable	Not applicable	Not applicable	Illegal
01011	Read with no intent to cache	Single-beat read or burst	Not applicable	Not applicable	Clean	Read, assert $\overline{\text{AACK}}$ and $\overline{\text{TA}}$
01111	Reserved	—	Not applicable	Not applicable	Not applicable	Illegal
1XX11	Reserved for customer	—	Not applicable	Not applicable	Not applicable	Illegal

[1] TT1 can be interpreted as a read-versus-write indicator for the bus.

[2] This column specifies the TT encoding for the general 60x protocol. The processor generates or snoops only a subset of those encodings.

NOTE

Regarding Table 8-2:

- For reads, the processor cleans or flushes during a snoop, based on the $\overline{\text{TBST}}$ input. The processor cleans for single-beat reads ($\overline{\text{TBST}}$ negated) to emulate read-with-no-intent-to-cache operations.
- Castouts and snoop copybacks are generally marked as non-global and are not snooped (except for reservation monitoring). However, other masters performing DMA write operations with the same TT encoding and marked as global WR operations (global or non-global) cancel an active reservation during a snoop hit in the reservation register (independent of a snoop hit in the cache).

- A non-processor read can cause the internal processor to invalidate the corresponding cache line if it exists.

8.4.3.2 Transfer Code Signals TC[0:2]

The transfer code signals, TC[0:2], provide supplemental information about the corresponding address (mainly regarding the source of the transaction). Note that TC*x* signals can be used with the TT[0:4] and $\overline{\text{TBST}}$ to further define the current transaction.

Table 8-3. Transfer Code Encoding for 60x Bus

TC[0:2]	60x Bus	
	Read	**Write**
000	Core data transaction	Any write
001	Core touch load	—
010	Core instruction fetch	—
011	Reserved	—
100	PCI bridge transaction	
101	SEC transaction	
110	DMA function code 0	
111	DMA function code 1	

8.4.3.3 $\overline{\text{TBST}}$ and TSIZ[0:3] Signals and Size of Transfer

As shown in Table 8-4, the transfer size signals (TSIZ[0:3]) and the transfer burst signal ($\overline{\text{TBST}}$) together indicate the size of the requested data transfer. These signals can be used with address bits A[27–31] and the device port size to determine which portion of the data bus contains valid data for a write transaction or which portion of the bus should contain valid data for a read transaction.

The MPC8272 uses four double-word burst transactions for transferring cache blocks. For these transactions, TSIZ[0–3] are encoded as 0b0010, and address bits A[27–28] determine which double word is sent first.

The MPC8272 supports critical-word-first burst transactions (double word–aligned) from the processor. The MPC8272 transfers the critical double word of data first, followed by the double words from increasing addresses, wrapping back to the beginning of the eight-word block as required.

Table 8-4. Transfer Size Signal Encoding

$\overline{\text{TBST}}$	TSIZ[0:3]	Transfer Size	Comments	Source
Negated	0001	1 byte	Byte	Core and DMA
Negated	0010	2 bytes	Half word	Core and DMA
Negated	0011	3 bytes	—	Core and DMA

Table 8-4. Transfer Size Signal Encoding (continued)

$\overline{\text{TBST}}$	TSIZ[0:3]	Transfer Size	Comments	Source
Negated	0100	4 bytes	Word	Core and DMA
Negated	0101	5 bytes	Extended 5 bytes	SDMA (MPC8272 only)
Negated	0110	6 bytes	Extended 6 bytes	SDMA (MPC8272 only)
Negated	0111	7 bytes	Extended 7 bytes	SDMA (MPC8272 only)
Negated	0000	8 bytes	Double word (maximum data bus size)	Core and DMA
Negated	1001	16 bytes	Extended double word	SDMA (MPC8272 only)
Negated	1010	24 bytes	Extended triple double word	SDMA (MPC8272 only)
Asserted	0010	32 bytes	Quad double word (4 maximum data beats)	Core and DMA

NOTE

The basic coherency size of the bus is 32 bytes for the processor (cache-block size). Data transfers that cross an aligned 32-byte boundary must present a new address onto the bus at that boundary for proper snoop operation, or must operate as non-coherent with respect to the MPC8272.

8.4.3.4 Burst Ordering During Data Transfers

During burst transfers, 32 bytes of data (one cache block) are transferred to or from the cache. Burst write transfers are performed zero double word–first. However, because burst reads are performed critical double word first, a burst-read transfer may not start with the first double word of the cache block, and the cache-block-fill operation may wrap around the end of the cache block. Table 8-5 describes the MPC8272 burst ordering.

Table 8-5. Burst Ordering

Data Transfer	Double-Word Starting Address:			
	A[27–28] = 00[1]	A[27–28] = 01	A[27–28] = 10	A[27–28] = 11
1st data beat	DW0[2]	DW1	DW2	DW3
2nd data beat	DW1	DW2	DW3	DW0
3rd data beat	DW2	DW3	DW0	DW1
4th data beat	DW3	DW0	DW1	DW2

[1] A[27–28] specifies the first double word of the 32-byte block being transferred; any subsequent double words must wrap-around the block. A[29–31] are always 0b000 for burst transfers by the MPC8272.

[2] DW*x* represents the double word that would be addressed by A[27–28] = *x* if a nonburst transfer were performed.

Each data beat is terminated with an assertion of $\overline{\text{TA}}$.

8.4.3.5 Effect of Alignment on Data Transfers

Table 8-6 lists the aligned transfers that can occur to and from the MPC8272. These are transfers in which the data is aligned to an address that is an integer multiple of the size of the data. For example, Table 8-6 shows that 1-byte data is always aligned; however, a 4-byte word must reside at an address that is a multiple of four to be aligned.

In Figure 8-6, Table 8-6, and Table 8-7, OP0 is the most-significant byte of a word operand and OP7 is the least-significant byte.

Table 8-6. Aligned Data Transfers

Program Transfer Size	TSIZ[0:3]	A[29:31]	Data Bus Byte Lanes							
			D0...			...D31	D32...			...D63
			B0	B1	B2	B3	B4	B5	B6	B7
Byte	0001	000	OP0[1]	—[2]	—	—	—	—	—	—
	0001	001	—	OP1	—	—	—	—	—	—
	0001	010	—	—	OP2	—	—	—	—	—
	0001	011	—	—	—	OP3	—	—	—	—
	0001	100	—	—	—	—	OP4	—	—	—
	0001	101	—	—	—	—	—	OP5	—	—
	0001	110	—	—	—	—	—	—	OP6	—
	0001	111	—	—	—	—	—	—	—	OP7
Half word	0010	000	OP0	OP1	—	—	—	—	—	—
	0010	010	—	—	OP2	OP3	—	—	—	—
	0010	100	—	—	—	—	OP4	OP5	—	—
	0010	110	—	—	—	—	—	—	OP6	OP7
Word	0100	000	OP0	OP1	OP2	OP3	—	—	—	—
	0100	100	—	—	—	—	OP4	OP5	OP6	OP7
Double word	0000	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7

[1] OP*n*: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand, and OP7 is the least-significant byte.

[2] —: These lanes are ignored during reads and driven with undefined data during writes.

The MPC8272 supports misaligned memory operations, although they may degrade performance substantially. A misaligned memory address is one that is not aligned to the size of the data being transferred (for example, a word read from an odd byte address). The MPC8272’s processor bus interface supports misaligned transfers within a word (32-bit aligned) boundary, as shown in Table 8-7. Note that the 4-byte transfer in Table 8-7 is only one example of misalignment. As long as the attempted transfer does not cross a word boundary, the MPC8272 can transfer the data to the misaligned address within a single bus transfer (for example, a half word read from an odd byte-aligned address). It takes two bus transfers to access data that crosses a word boundary.

Due to the performance degradation, misaligned memory operations should be avoided. In addition to the double-word straddle boundary condition, the processor’s address translation logic can generate substantial exception overhead when the load/store multiple and load/store string instructions access misaligned data. It is strongly recommended that software attempt to align code and data where possible.

Table 8-7. Unaligned Data Transfer Example (4-Byte Example)

Program Size of Word (4 bytes)	TSIZ[1:3]	A[29:31]	Data Bus Byte Lanes							
			D0...			...D31	D32...			...D63
			B0	B1	B2	B3	B4	B5	B6	B7
Aligned	100	000	A[1]	A	A	A	—[2]	—	—	—
Misaligned—1st access	011	001	—	A	A	A	—	—	—	—
2nd access	001	100	—	—	—	—	A	—	—	—
Misaligned—1st access	010	010	—	—	A	A	—	—	—	—
2nd access	010	100	—	—	—	—	A	A	—	—
Misaligned—1st access	001	011	—	—	—	A	—	—	—	—
2nd access	011	100	—	—	—	—	A	A	A	—
Aligned	100	100	—	—	—	—	A	A	A	A
Misaligned—1st access	011	101	—	—	—	—	—	A	A	A
2nd access	001	000	A	—	—	—	—	—	—	—
Misaligned—1st access	010	110	—	—	—	—	—	—	A	A
2nd access	010	000	A	A	—	—	—	—	—	—
Misaligned—1st access	001	111	—	—	—	—	—	—	—	A
2nd access	011	000	A	A	A	—	—	—	—	—

[1] A: Byte lane used.

[2] —: Byte lane not used.

8.4.3.6 Effect of Port Size on Data Transfers

The MPC8272 can transfer operands through its 64-bit data port. If the transfer is controlled by the internal memory controller, the MPC8272 can support 8-, 16-, 32-, and 64-bit data port sizes as demonstrated in Figure 8-6. The bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 64-bit port must reside on data bus bits D[0–63], a 32-bit port must reside on bits D[0–31], a 16-bit port must reside on bits D[0–15], and an 8-bit port must reside on bits D[0–7]. The MPC8272 always tries to transfer the maximum amount of data on all bus cycles. For a word operation, it always assumes the port is 64 bits wide when beginning the bus cycle; for burst and extended byte cycles, a 64-bit bus is assumed.

Figure 8-6 shows the device connections on the data bus. Table 8-8 lists the bytes required on the data bus for read cycles.

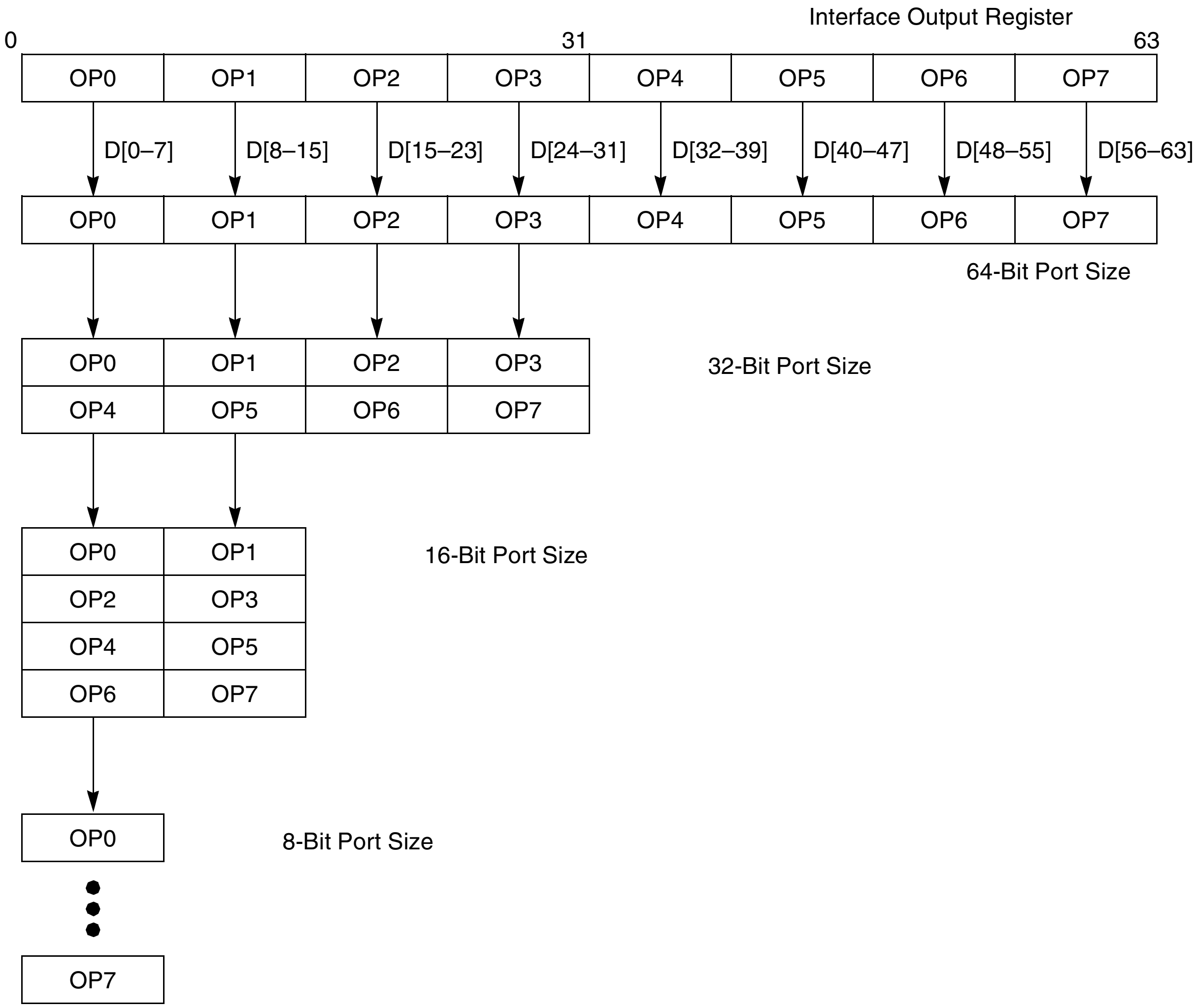


Figure 8-6. Interface to Different Port Size Devices

Table 8-8. Data Bus: Read Cycle Requirements and Write Cycle Content

Transfer Size TSIZ[0:3]	Address State [1] A[29:31]	Port Size/Data Bus Assignments														
		64-Bit								32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	32–39	40–47	48–55	56–63	0–7	8–15	16–23	24–31	0–7	8–15	0–7
Byte (0001)	000	OP0 [2]	—[3]	—	—	—	—	—	—	OP0	—	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	—	—	—	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	—	—	—	—	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	—	—	—	—	—	—	OP3	—	OP3	OP3
	100	—	—	—	—	OP4	—	—	—	OP4	—	—	—	OP4	—	OP4
	101	—	—	—	—	—	OP5	—	—	—	OP5	—	—	—	OP5	OP5
	110	—	—	—	—	—	—	OP6	—	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	—	—	—	—	OP7	—	—	—	OP7	—	OP7	OP7
Half word (0010)	000	OP0	OP1	—	—	—	—	—	—	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	—	—	—	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	—	—	—	—	—	—	OP2	OP3	OP2	OP3	OP2
	100	—	—	—	—	OP4	OP5	—	—	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	—	—	—	—	OP5	OP6	—	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	—	—	—	—	OP6	OP7	—	—	OP6	OP7	OP6	OP7	OP6
Triple Byte (0011)	000	OP0	OP1	OP2	—	—	—	—	—	OP0	OP1	OP2	—	OP0	OP1	OP0
	001	—	OP1	OP2	OP3	—	—	—	—	—	OP1	OP2	OP3	—	OP1	OP1
	100	—	—	—	—	OP4	OP5	OP6	—	OP4	OP5	OP6	—	OP4	OP5	OP4
	101	—	—	—	—	—	OP5	OP6	OP7	—	OP5	OP6	OP7	—	OP5	OP5
Word (0100)	000	OP0	OP1	OP2	OP3	—	—	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	—	—	—	—	OP4	OP5	OP6	OP7	OP4	OP5	OP6	OP7	OP4	OP5	OP4
Double Word (0000)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7	OP0	OP1	OP2	OP3	OP0	OP1	OP0

[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 OP7 is the least-significant byte.

[3] —: These lanes are ignored during read cycles and driven with undefined data during write cycles.

8.4.3.7 60x-Compatible Bus Mode—Size Calculation

To comply with the requirements listed in Table 8-6 and Table 8-7, the transfer size and a new address must be calculated at the termination of each beat of a port-size transaction. In single-MPC8272 bus mode, these calculations are internal and do not constrain the system. In 60x-compatible bus mode, the external slave or master must determine the new address and size. Table 8-9 describes the address and size calculation state machine. Note that the address and size states are for internal use and are not transferred on the address or TSIZ pins. Extended transactions (16- and 24-byte) are not described here but can be determined by extending this table for 9-, 10-, 16-, 23-, and 24-byte transactions.

Table 8-9. Address and Size State Calculations

Size State	Address State [0–4]					Port Size	Next Size State	Next Address State [0–4]				
Byte	x	x	x	x	x	x	Stop					
2-byte	x	x	x	x	0	Byte	Byte	x	x	x	x	1
	x	x	0	0	1		Byte	x	x	0	1	0
	x	x	1	0	1		Byte	x	x	1	1	0
	x	x	x	0	1	Half	Byte	x	x	x	1	0
	x	x	x	x	0		Stop					
3-byte	x	x	0	0	0	Byte	2-byte	x	x	0	0	1
	x	x	0	0	1		2-byte	x	x	0	1	0
	x	x	1	0	0		2-byte	x	x	1	0	1
	x	x	1	0	1		2-byte	x	x	1	1	0
	x	x	0	0	0	Half	Byte	x	x	0	1	0
	x	x	0	0	1		2-byte	x	x	0	1	0
	x	x	1	0	0		Byte	x	x	1	1	0
	x	x	1	0	1		2-byte	x	x	1	1	0
	x	x	x	x	x	Word	Stop					
4-byte	x	x	x	0	0	Byte	3-byte	x	x	x	0	1
	x	x	x	0	0	Half	2-byte	x	x	x	1	0
	x	x	x	x	x	Word	Stop					
5-byte	x	x	0	1	1	Byte	4-byte	x	x	1	0	0
6-byte	x	x	0	1	0	Byte	5-byte	x	x	0	1	1
	x	x	0	1	0	Half	4-byte	x	x	1	0	0
7-byte	x	x	0	0	1	Byte	6-byte	x	x	0	1	0
8-byte	x	x	0	0	0	Byte	7-byte	x	x	0	0	1
	x	x	0	0	0	Half	6-byte	x	x	0	1	0
	x	x	0	0	0	Word	4-byte	x	x	1	0	0
	x	x	0	0	0	Double	Stop					

8.4.3.8 Extended Transfer Mode

The MPC8272 supports an extended transfer mode that improves bus performance. This should not be confused with the extended bus protocol used to support direct-store operations supported in some earlier processors that implement the PowerPC architecture. The MPC8272 can generate 5-, 6-, 7-, 16-, or 24-byte extended transfers. These transactions are compatible with the 60x bus, but some slaves or masters do not support these features. Clear BCR[ETM] to disable this type of transaction. This places the MPC8272 in strict 60x bus mode. The following tables are extensions to Table 8-7, Table 8-8, and Table 8-9.

Table 8-10 lists the patterns of the extended data transfer for write cycles when MPC8272 initiates an access. Note that 16- and 24-byte transfers are always 8-byte aligned and use a 64-bit or less port size.

Table 8-10. Data Bus Contents for Extended Write Cycles

Transfer Size TSIZ[0–3])	Address State A[29–31]	External Data Bus Pattern							
		D[0–7]	D[8–15]	D[16–23]	D[24–31]	D[32–39]	D[40–47]	D[48–55]	D[56–63]
5 bytes (0101)	000	OP0	OP1	OP2	OP3	OP4	—	—	—
	011	OP3	OP3	—	OP3	OP4	OP5	OP6	OP7
6 bytes (0110)	000	OP0	OP1	OP2	OP3	OP4	OP5	—	—
	010	OP2	OP3	OP2	OP3	OP4	OP5	OP6	OP7
7 bytes (0111)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	—
	001	OP1	OP1	OP2	OP3	OP4	OP5	OP6	OP7

Table 8-11 lists the bytes required on the data bus for extended read cycles. Note that 16- and 24-byte transfers are always 8-byte aligned and use a maximum 64-bit port size.

Table 8-11. Data Bus Requirements for Extended Read Cycles

Transfer Size TSIZ[0–3]	Address State A[29-31]	Port Size/Data Bus Assignments														
		64-Bit								32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	32–39	40–47	48–55	56–63	0–7	8–15	16–23	24–31	0–7	8–15	0–7
5 byte (0101)	000	OP0	OP1	OP2	OP3	OP4	—	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	011	—	—	—	OP3	OP4	OP5	OP6	OP7	—	—	—	OP3	—	OP3	OP3
6 byte (0110)	000	OP0	OP1	OP2	OP3	OP4	OP5	—	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	010	—	—	OP2	OP3	OP4	OP5	OP6	OP7	—	—	OP2	OP3	OP2	OP3	OP2
7 byte (0111)	000	OP0	OP1	OP2	OP3	OP4	OP5	OP6	—	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	001	—	OP1	OP2	OP3	OP4	OP5	OP6	OP7	—	OP1	OP2	OP3	—	OP1	OP1

Table 8-12 includes added states to the transfer size calculation state machine. Only extended transfers use these states.

Table 8-12. Address and Size State for Extended Transfers

Size State [0–3]	Address State[0–4]					Port Size	Next Size State [0–3]	Next Address State[0–4]				
Half	x	x	x	1	1	Byte	Byte	x	x	1	0	0
	x	x	1	0	1			x	x	1	1	0
	x	x	x	x	x	Half	Stop					
3-byte	x	x	0	1	0	Byte	Half	x	x	0	1	1
	x	x	1	0	0			x	x	1	0	1
	x	x	0	1	0	Half	Byte	x	x	1	0	0
	x	x	1	0	0			x	x	1	1	0
Word	x	x	0	0	1	Byte	3-byte	x	x	0	1	0
	x	x	0	1	1			x	x	1	0	0
5-byte	x	x	0	0	0	Byte	Word	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	1	0			x	x	0	1	1
	x	x	0	1	1			x	x	1	0	0
	x	x	0	0	0	Half	3-byte	x	x	0	1	0
	x	x	0	1	0			x	x	1	0	0
	x	x	0	1	1		Word	x	x	1	0	0
	x	x	0	0	0	Word	Byte	x	x	1	0	0
	x	x	0	1	1		Word	x	x	1	0	0
	x	x	x	x	x	Double	Stop					
6-byte	x	x	0	0	0	Byte	5-byte	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	1	0			x	x	0	1	1
	x	x	0	0	0	Half	Word	x	x	0	1	0
	x	x	0	1	0			x	x	1	0	0
	x	x	0	0	0	Word	Half	x	x	1	0	0
	x	x	0	1	0		Word	x	x	1	0	0
	x	x	x	x	x	Double	Stop					

Table 8-12. Address and Size State for Extended Transfers (continued)

Size State [0–3]	Address State[0–4]					Port Size	Next Size State [0–3]	Next Address State[0–4]				
7-byte	x	x	0	0	0	Byte	6-byte	x	x	0	0	1
	x	x	0	0	1			x	x	0	1	0
	x	x	0	0	0	Half	5-byte	x	x	0	1	0
	x	x	0	0	1		6-byte	x	x	0	1	0
	x	x	0	0	0	Word	3-byte	x	x	1	0	0
	x	x	0	0	1		4-byte	x	x	1	0	0
	x	x	x	x	x	Double	Stop					

Extended transfer mode is enabled by setting the BCR[ETM].

8.4.4 Address Transfer Termination

Address transfer termination occurs with the assertion of the address acknowledge (AACK) signal or address retry ARTRY signal. ARTRY must remain asserted until 1 clock after AACK; the bus clock cycle after AACK is called the ARTRY window. The MPC8272 controls assertion of AACK unless the cycle is claimed by an external slave. When AACK is asserted by the external slave, it should be asserted for 1 clock cycle and negated for 1 clock cycle before entering a high-impedance state. The MPC8272 holds AACK in a high-impedance state until it is required to assert AACK to terminate the address cycle.

The MPC8272 uses AACK to enforce a pipeline depth of one to its internal slaves.

8.4.4.1 Address Retried with ARTRY

The address transfer can be terminated with the requirement to retry if ARTRY is asserted during the address tenure and through the cycle following AACK. The assertion causes the entire transaction (address and data tenure) to be rerun. As a snooping device, the MPC8272 processor asserts ARTRY for a snooped transaction that hits modified data in the data cache that must be written back to memory, or if the snooped transaction could not be serviced. As a bus master, the MPC8272 responds to an assertion of ARTRY by aborting the bus transaction and requesting the bus again, as shown in Figure 8-7. Note that after recognizing an assertion of ARTRY and aborting the current transaction, the MPC8272 may not run the same transaction the next time it is granted the bus.

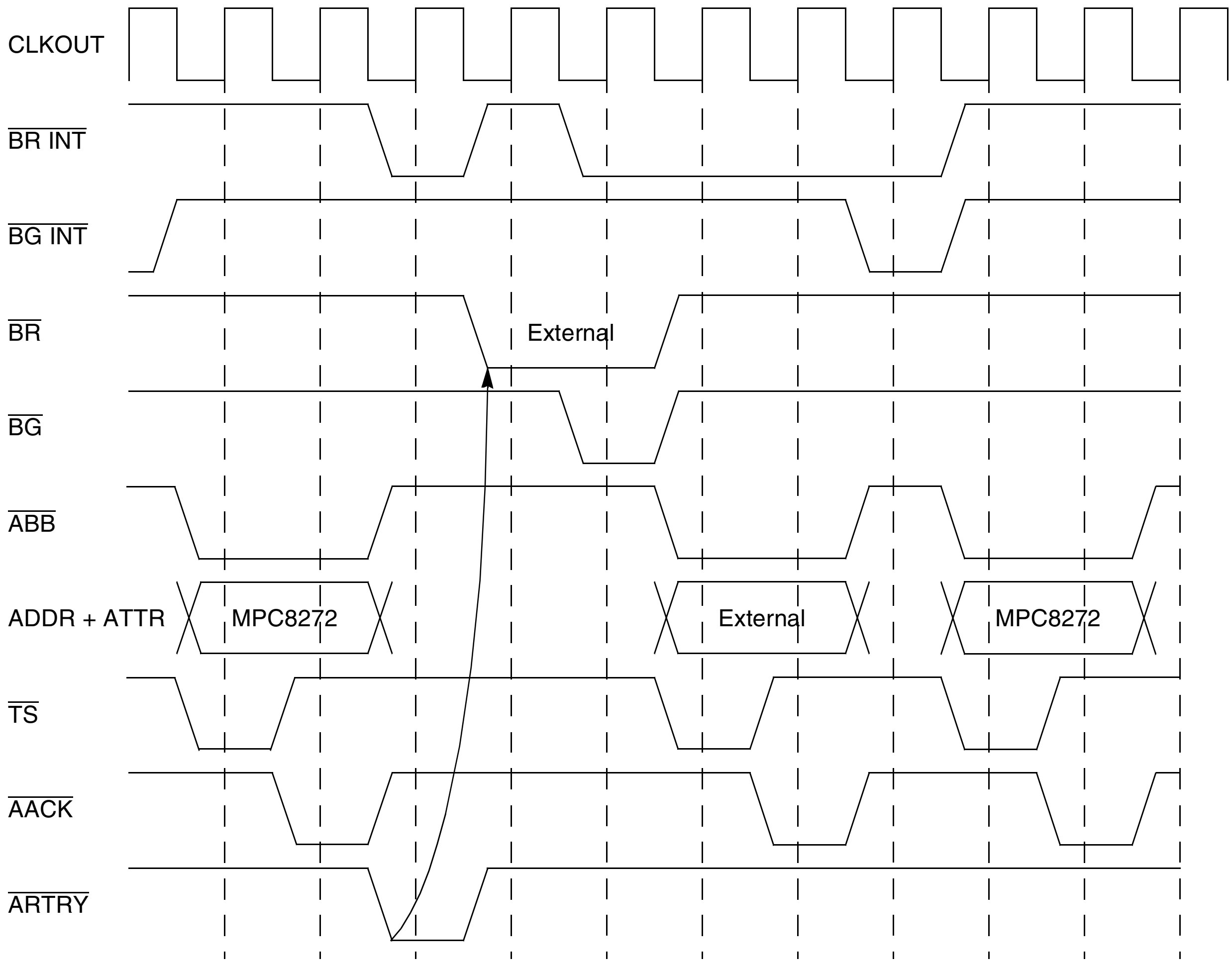


Figure 8-7. Retry Cycle

As a bus master, the MPC8272 recognizes either an early or qualified $\overline{\text{ARTRY}}$ and prevents the data tenure associated with the retried address tenure. If the data tenure has begun, the MPC8272 terminates the data tenure immediately even if the burst data has been received. If the assertion of $\overline{\text{ARTRY}}$ is received up to or on the bus cycle as the first (or only) assertion of $\overline{\text{TA}}$ for the data tenure, the MPC8272 ignores the first data beat. If it is a read operation, the MPC8272 does not forward data internally to the cache, execution unit, or any other MPC8272 internal storage. This address retry case succeeds because the data tenure is aborted in time, and the entire transaction is rerun. This retry mechanism allows the memory system to begin operating in parallel with the bus snoopers, provided external devices do not present data sooner than the bus cycle before all snoop responses can be determined and asserted on the bus.

Note that the system must ensure that $\overline{\text{ARTRY}}$ is never asserted later than the cycle of the first or only assertion of $\overline{\text{TA}}$ (if the PCI controller can initiate global transactions, the system must ensure that ARTRY is never asserted on the same cycle or later than the first or only assertion of $\overline{\text{TA}}$). This guarantees the relationship between $\overline{\text{TA}}$ and $\overline{\text{ARTRY}}$ such that, in case of an address retry, the data can be cancelled in the chip before it can be forwarded internally to the internal memory resources (registers or cache). Generally, the memory system must also detect this event and abort any transfer in progress. If this $\overline{\text{TA}}/\overline{\text{ARTRY}}$

relationship is not met, the master may enter an undefined state. Users may use PPC_ACR[DBGD] to ensure correct operation of the system.

During the clock of a qualified $\overline{\text{ARTRY}}$, each device master determines whether it should negate $\overline{\text{BR}}$ and ignore $\overline{\text{BG}}$ on the following cycle. The following cycle is referred to as the window-of-opportunity for the snooping master. During this window, only the snooping master that asserted $\overline{\text{ARTRY}}$ and requires a snoop copyback operation is allowed to assert $\overline{\text{BR}}$. This guarantees the snooping master a window of opportunity to request and be granted the bus before the just-retried master can restart its transaction. $\overline{\text{BG}}$ is also blocked in the window-of-opportunity, so the arbiter has a chance to negate $\overline{\text{BG}}$ to an already granted potential bus master to perform a new arbitration.

Note that in some systems, an external processor may be unable to perform a pending snoop copyback when a new snoop operation is performed. In this case, the MPC8272 requests the window of opportunity if it hits on the new snooped address. To clear its internal snoop queue, it performs the snoop copyback operation for the earlier snooped address instead of the current snooped address.

8.4.4.2 Address Tenure Timing Configuration

During address tenures initiated by 60x-bus devices, the timing of the assertion of $\overline{\text{AACK}}$ by the MPC8272 is determined by the BCR[APD] and the pipeline status of the 60x bus. Because the MPC8272 can support one level of pipelining, it uses $\overline{\text{AACK}}$ to control the 60x-bus pipeline condition. To maintain the one-level pipeline, $\overline{\text{AACK}}$ is not asserted for a pipelined address tenure until the current data tenure ends. The MPC8272 also delays asserting $\overline{\text{AACK}}$ until no more address retry conditions can occur. Note that the earliest the MPC8272 can assert $\overline{\text{AACK}}$ is the clock cycle when the wait-state values set by BCR[APD] have expired.

BCR[APD] specifies the minimum number of address tenure wait states for address operations initiated by 60x-bus devices. APD indicates how many cycles the MPC8272 should wait for $\overline{\text{ARTRY}}$, but because it is assumed that $\overline{\text{ARTRY}}$ can be asserted (by other masters) only on cacheable address spaces, APD is considered only on transactions that hit a 60x-assigned memory controller bank and that have $\overline{\text{GBL}}$ asserted during the address phase.

Extra wait states may occur because of other MPC8272 configuration parameters. In systems with multiple potential masters, the number of wait states configured by BCR[APD] should be at least as large as the value the slowest master would need to assert a snoop response. For example, additional wait states are required when the internal processor is in 1:1 clock mode; this case requires at least one wait state to generate the $\overline{\text{ARTRY}}$ response.

8.4.5 Pipeline Control

The MPC8272 supports the two following modes:

- One-level pipeline mode—To maintain the one-level pipeline, $\overline{\text{AACK}}$ is not asserted for a pipelined address tenure until the current data tenure ends. In 60x-compatible bus mode, a two-level pipeline depth can occur (for example, when an external 60x-bus slave does not support one-level pipelining). When the internal arbiter counts a pipeline depth of two (two assertions of $\overline{\text{AACK}}$ before the assertion of the current data tenure) it negates all address bus grant ($\overline{\text{BG}}$) signals.

- No-pipeline mode—The MPC8272 does not assert AACK until the corresponding data tenure ends.

8.5 Data Tenure Operations

This section describes the operation of the MPC8272 during the data bus arbitration, transfer, and termination phases of the data tenure.

NOTE:

> For external master writes to DPRAM, DPRAM is clocked by the CPM clock, not the 60x bus clock. Therefore, data is not latched at the TA assertion cycle during writes to DPRAM from the external master; rather, it is latched earlier. It is necessary, then, that the external master drive the data bus immediately after DBG and hold the data bus until after TA.

8.5.1 Data Bus Arbitration

The beginning of an address transfer, marked by the assertion of transfer start (TS), is also an implicit data bus request provided that the transfer type signals (TT[0:4]) indicate that the transaction is not address-only.

The MPC8272 arbiter supports one external master and uses DBG to grant the external master data bus.The DBG signals are not asserted if the data bus, which is shared with memory, is busy with a transaction.

A qualified data bus grant (QDBG) can be expressed as the assertion of DBG while DBB and ARTRY (associated with the data bus operation) are negated.

Note that the MPC8272 arbiter should assert DBG only when it is certain that the first TA will be asserted with or after the associated ARTRY. The MPC8272 DBG is asserted with TS if the data bus is free and if the PPC_ACR[DBGD] = 0. If PPC_ACR[DBGD] = 1, DBG is asserted one cycle after TS if the data bus is not busy. The DBG delay should be used to ensure that ARTRY is not asserted after the first or only TA assertion. For the programming model, see Section 4.3.2.2, "60x Bus Arbiter Configuration Register (PPC_ACR)."

Note that DBB should not be asserted after the data tenure is finished. Assertion of DBB after the last TA causes improper operation of the bus. (MPC8272 internal masters do not assert DBB after the last TA.)

Note the following:

- External bus arbiters must comply with the following restriction on assertion of DBG, which is connected to the MPC8272. In case the data bus is not busy with the data of a previous transaction on the bus, the external arbiter must assert DBG in the same cycle in which TS is asserted (by a master that was granted the bus) or in the following cycle. In case the external arbiter asserts DBG on the cycle in which TS was asserted, PPC_ACR[DBGD] should be zero. Otherwise, PPC_ACR[DBGD] should be set.
- External masters connected to the 60x bus must assert DBB only for the duration of its data tenure. External masters should not use DBB to prevent other masters from using the data bus after their data tenure has ended.

8.5.2 Data Streaming Mode

The MPC8272 supports a special data streaming mode that can improve bus performance in some conditions. Generally, the bus protocol requires one idle cycle between any two data tenures. This idle cycle is essential to prevent contention on the data bus when the driver of the data is changing. However, when the driver on the data bus is the same for both data tenures, this idle cycle may be omitted.

In data streaming mode, the MPC8272 omits the idle cycle where possible. MPC8272 applications often require data stream transfers of more than 4 x 64 bits. For example, the ATM cell's payload is 6 x 64 bits. All this data is driven from a single device on the bus, so data streaming saves a cycle for such a transfer. When data-streaming mode is enabled, transactions initiated by the core are not affected, while transactions initiated by other bus masters within the chip omit the idle cycle if the data driver is the same. Note that data streaming mode cannot be enabled when the MPC8272 is in 60x-compatible bus mode and a device that uses DBB is connected to the bus. This restriction is due to the fact that a MPC8272 for which data streaming mode is enabled may leave DBB asserted after the last TA of a transaction, which is a violation of the strict bus protocol. The data streaming mode is enabled by setting BCR[ETM].

8.5.3 Data Bus Transfers and Normal Termination

The data transfer signals include D[0:63]. For memory accesses, the data signals form a 64-bit data path, D[0–63], for read and write operations.

The MPC8272 handles data transfers in either single-beat or burst operations. Single-beat operations can transfer from 1 to 24 bytes of data at a time. Burst operations always transfer eight words in four double-word beats. A burst transaction is indicated by the assertion of TBST by the bus master. A transaction is terminated normally by asserting TA.

The three following signals are used to terminate the individual data beats of the data tenure and the data tenure itself:

- TA indicates normal termination of data transactions. It must always be asserted on the bus cycle coincident with the data that it is qualifying. It may be withheld by the slave for any number of clocks until valid data is ready to be supplied or accepted.
- Asserting TEA indicates a nonrecoverable bus error event. Upon receiving a final (or only) termination condition, the MPC8272 always negates DBB for 1 cycle, except when a fast data bus grant is performed.
- Asserting ARTRY causes the data tenure to be terminated immediately if the ARTRY is for the address tenure associated with the data tenure in operation (the data tenure may not be terminated due to address pipelining). The earliest allowable assertion of TA depends directly on the latest possible assertion of ARTRY.

Figure 8-8 shows both a single-beat and burst data transfer. The MPC8272 asserts TA to mark the cycle in which data is accepted. In a normal burst transfer, the fourth assertion of TA signals the end of a transfer.

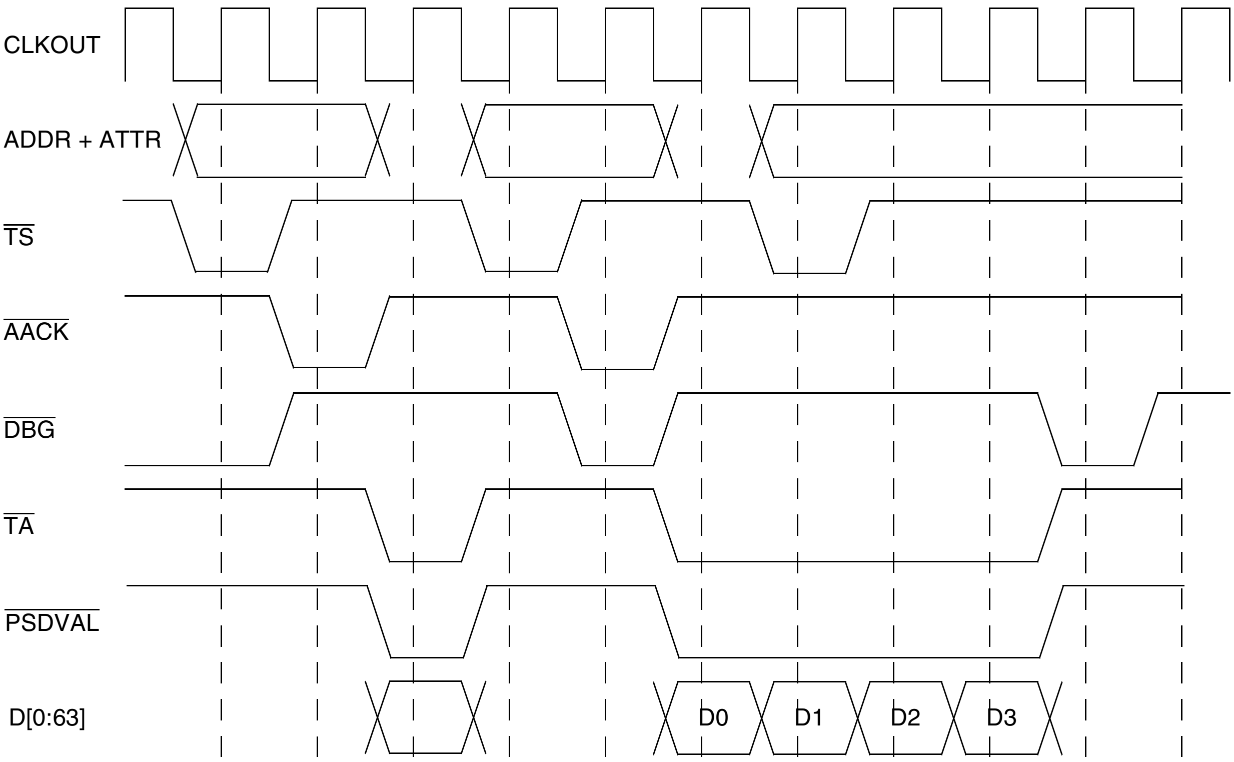


Figure 8-8. Single-Beat and Burst Data Transfers

8.5.4 Effect of ARTRY Assertion on Data Transfer and Arbitration

The MPC8272 allows an address tenure to overlap its associated data tenure. The MPC8272 internally guarantees that the first TA of the data tenure is delayed to be at the same time or after the ARTRY window (the clock after the assertion of AACK).

8.5.5 Port Size Data Bus Transfers and PSDVAL Termination

The MPC8272 can transfer data through data ports of 8, 16, 32, and 64 bits, as shown in Section 8.4.3, “Address Transfer Attribute Signals.” Single-beat transaction sizes can be 8, 16, 32, 64, 128, and 192 bits; burst transactions are 256 bits. Single-beat and burst transactions are divided into to a number of intermediate beats depending on the port size. The MPC8272 asserts PSDVAL to mark the cycle in which data is accepted. Assertion of PSDVAL in conjunction with TA marks the end of the transfer in single-beat mode. The fourth assertion of PSDVAL in conjunction with TA signals the end of a burst transfer. Figure 8-9 shows an extended transaction of 4 words to a port size of 32 bits. The single-beat transaction is translated to four port-sized beats.

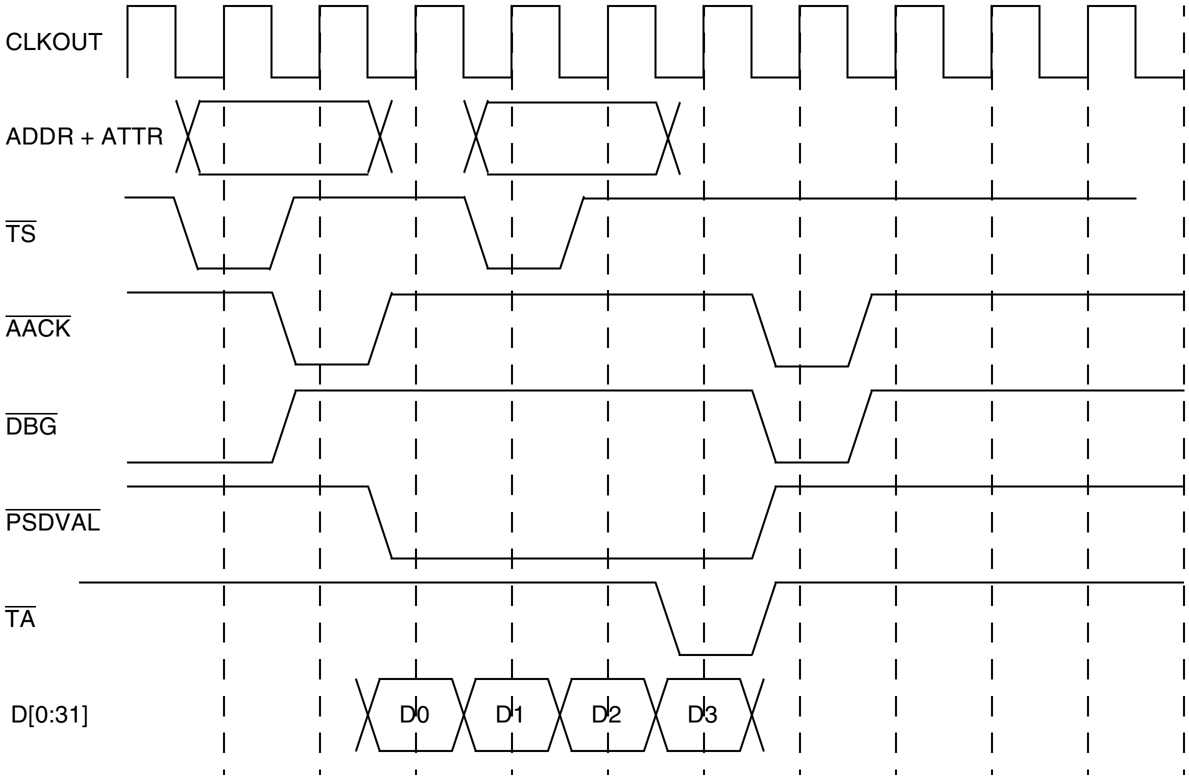


Figure 8-9. 28-Bit Extended Transfer to 32-Bit Port Size

Figure 8-10 shows a burst transfer to a 32-bit port. Each double-word burst beat is divided into two port-sized beats such that the four double words are transferred in eight beats.

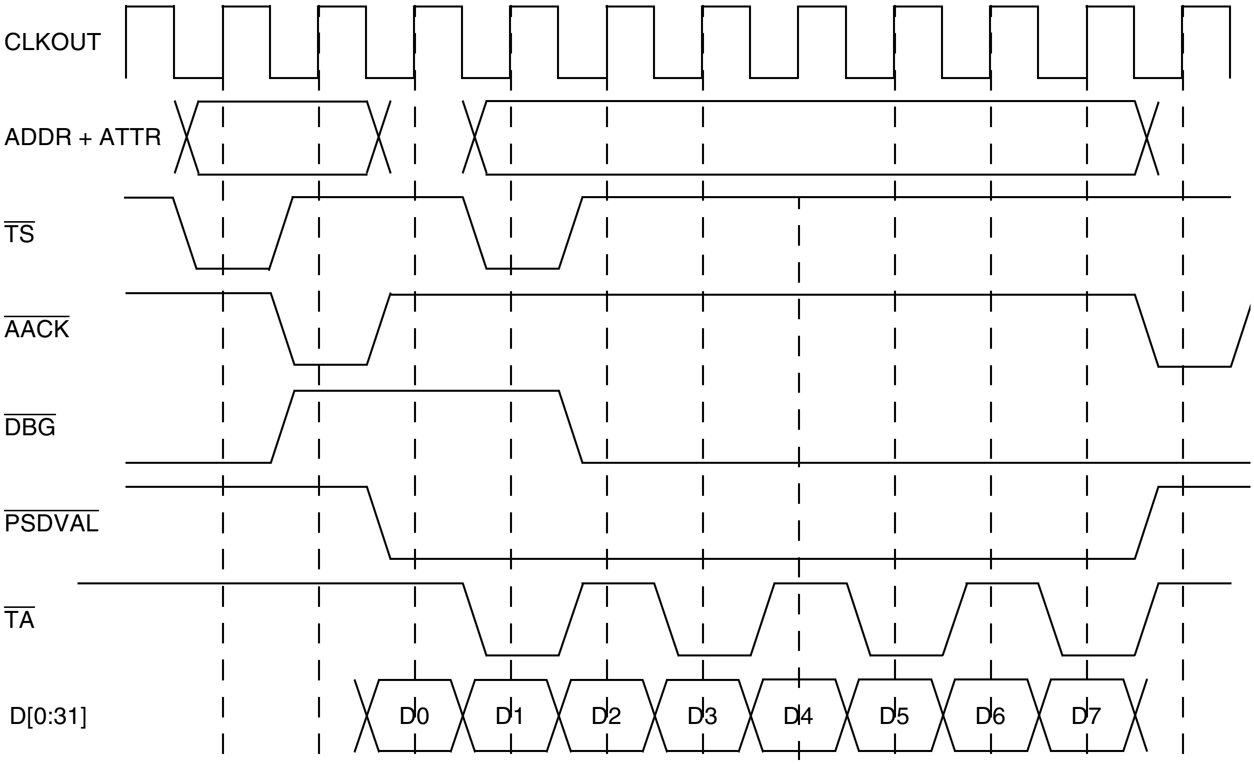


Figure 8-10. Burst Transfer to 32-Bit Port Size

8.5.6 Data Bus Termination by Assertion of TEA

If a device initiates a transaction that is not supported by the MPC8272, the MPC8272 signals an error by asserting TEA. Because the assertion of TEA is sampled by the device only during the data tenure of the bus transaction, the MPC8272 ensures that the device master receives a qualified data bus grant by asserting DBG before asserting TEA. The data tenure is terminated by a single assertion of TEA regardless of the port size or whether the data tenure is a single-beat or burst transaction. This sequence is shown in Figure 8-11. In Figure 8-11 the data bus is busy at the beginning of the transaction, thus delaying the assertion of DBG.

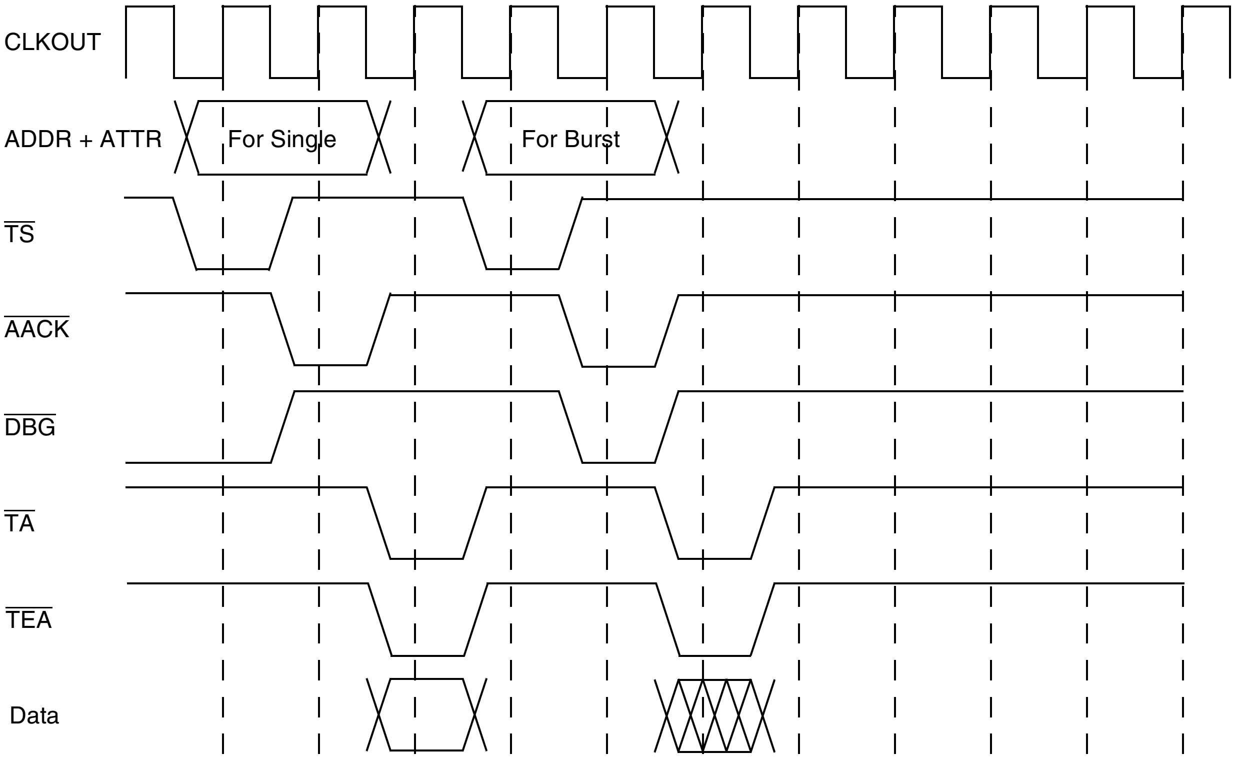


Figure 8-11. Data Tenure Terminated by Assertion of $\overline{TEA}$

The MPC8272 interprets the following bus transactions as bus errors:

- Direct-store transactions, as indicated by the assertion of $\overline{XATS}$
- Bus errors asserted by slaves (internal or external)

8.6 Memory Coherency—MEI Protocol

The MPC8272 provides dedicated hardware to ensure memory coherency by snooping bus transactions and maintaining information about the status of data in a cache block, and through the address retry capability. Each data cache block includes status bits that support the modified/exclusive/invalid, or MEI, cache-coherency protocol.

Asserting the global ($\overline{GBL}$) output signal indicates whether the current transaction must be snooped by other snooping devices on the bus. Address bus masters assert $\overline{GBL}$ to indicate that the current transaction is a global access (that is, an access to memory shared by more than one device). If $\overline{GBL}$ is not asserted for the transaction, that transaction is not snooped. When other devices detect the $\overline{GBL}$ input asserted, they must respond by snooping any addresses broadcast. Normally, $\overline{GBL}$ reflects the M bit value specified for the memory reference in the corresponding translation descriptor. Care must be taken to minimize the number of pages marked as global because the retry protocol discussed in the previous section used to enforce coherency can require significant bus bandwidth.

When the MPC8272 processor is not the address bus master, $\overline{GBL}$ is an input. The MPC8272 processor snoops a transaction if $\overline{TS}$ and $\overline{GBL}$ are asserted together in the same bus clock cycle (a qualified snooping condition). No snoop update to the MPC8272 processor cache occurs if the transaction is not marked global. This includes invalidation cycles.

When the MPC8272 processor detects a qualified snoop condition, the address associated with the $\overline{\text{TS}}$ is compared against the data cache tags. Snooping completes if no hit is detected. However, if the address hits in the cache, the MPC8272 processor reacts according to the MEI protocol shown in Figure 8-12. This figure assumes that WIM = 0b001 (memory space is marked for write-back, caching-allowed, and coherency-enforced modes).

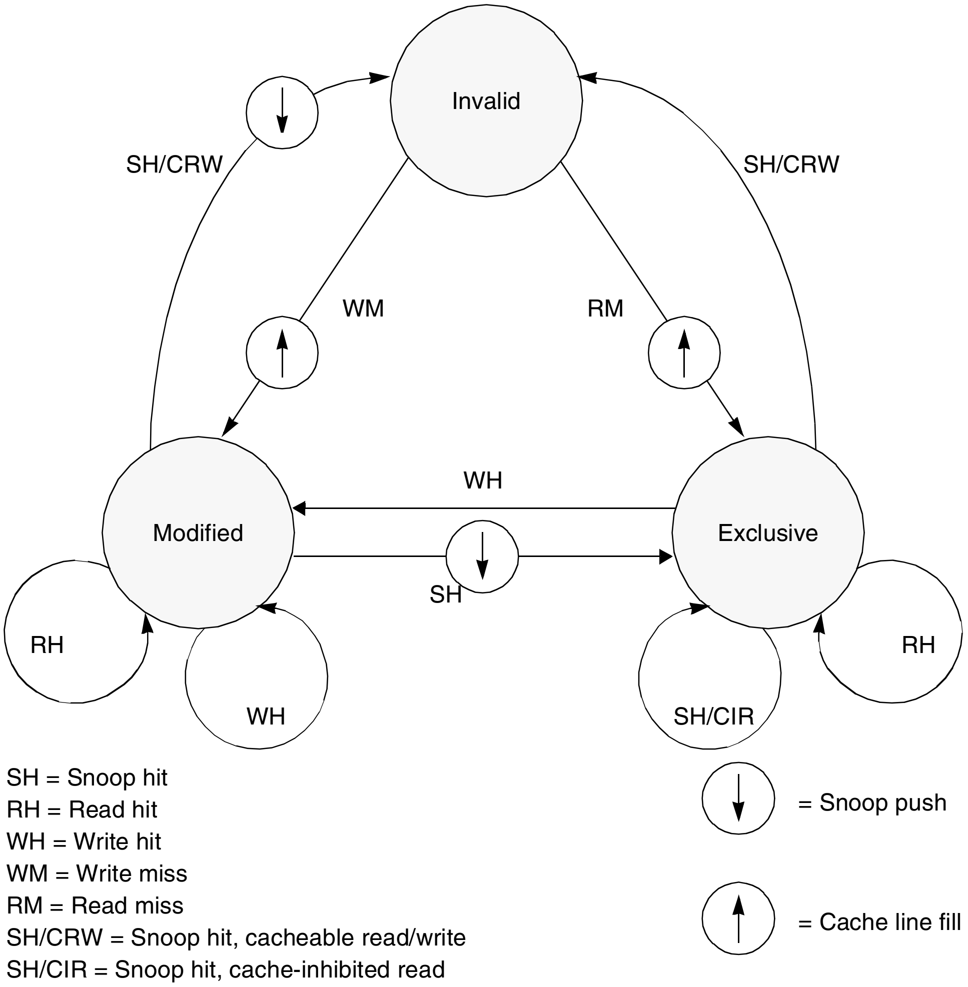


Figure 8-12. MEI Cache Coherency Protocol—State Diagram (WIM = 001)

8.7 Processor State Signals

This section describes the MPC8272's support for atomic update and memory through the use of the **lwarx**/**stwcx.** instruction pair. It also describes the $\overline{\text{TLBISYNC}}$ input.

8.7.1 Support for the lwarx/stwcx. Instruction Pair

The load word and reserve indexed (**lwarx**) and the store word conditional indexed (**stwcx.**) instructions provide a way to update memory atomically by setting a reservation on the load and checking that the reservation is still valid before the store is performed. In the MPC8272, reservations are made on behalf of aligned, 32-byte sections of the memory address space.

The reservation (RSRV) output signal is driven synchronously with the bus clock and reflects the status of the reservation coherency bit in the reservation address register.

Note that each external master must do its own snooping; the MPC8272 does not provide external reservation snooping.

8.7.2 TLBISYNC Input

The TLBISYNC input permits hardware synchronization of changes to MMU tables when the MPC8272 and another DMA master share the MMU translation tables in system memory. A DMA master asserts TLBISYNC when it uses shared addresses that the MPC8272 can change in the MMU tables during the DMA master's tenure.

When the TLBISYNC input is asserted, the MPC8272 cannot complete any instructions past a **tlbsync** instruction. Generally, during the execution of an **eciwx** or **ecowx** instruction, the selected DMA device should assert the MPC8272's TLBISYNC signal and hold it asserted during its DMA tenure if it is using a shared translation address. Subsequent instructions by the MPC8272 processor should include a **sync** and **tlbsync** instruction before any MMU table changes are performed. This prevents the MPC8272 from making disruptive table changes during the DMA tenure.

8.8 Little-Endian Mode

The MPC8272 supports a little-endian mode in which low-order address bits are operated on (munged) based on the size of the requested data transfer. This mode allows a little-endian program running on the processor with a big-endian memory system to offset into a data structure and receive the same results as it would if it were operating on a true little-endian processor and memory system (for example, writing a word to memory as a word operation on the bus and then reading in the second byte of that word as a byte operation on the bus).

NOTE

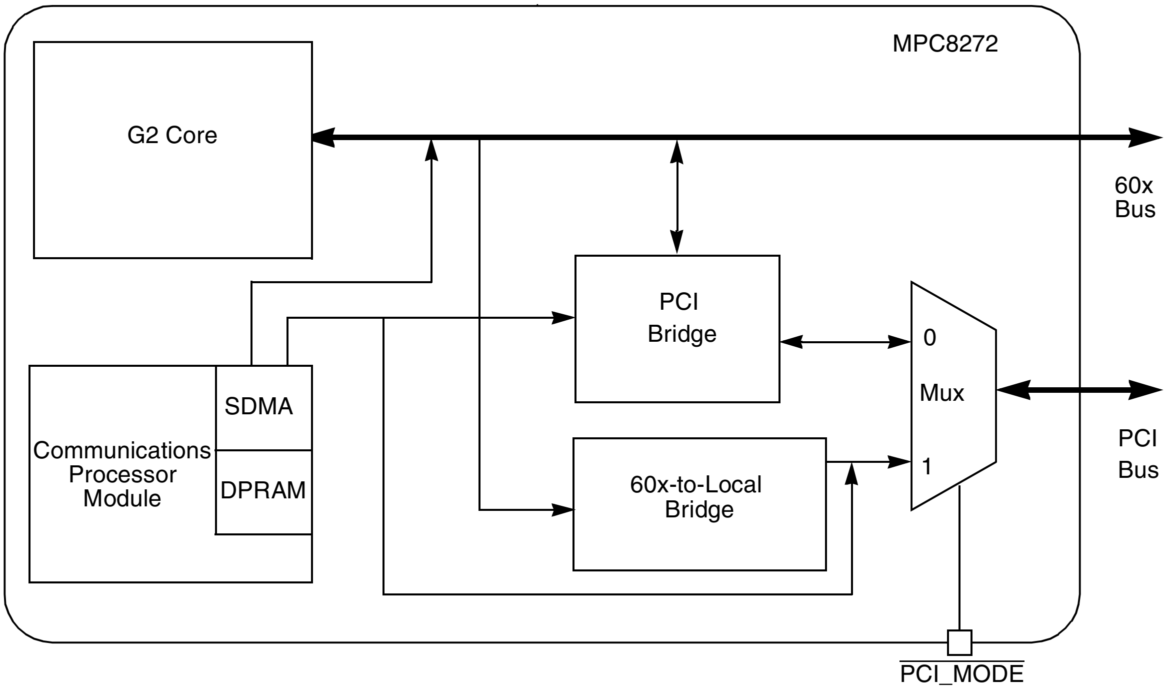
When the processor is selected for little-endian operation, the bus interface is still operating in big-endian mode. That is, byte address 0 of a double word (as selected by A[29–31] on the bus—after the internal address munge) still selects the most significant (left most) byte of the double word on D[0–7]. If the processor interfaces with a true little-endian environment, the system may need to perform byte-lane swapping or other operations external to the processor.

Chapter 9
PCI Bridge

The PCI bridge enables the MPC8272 to gluelessly bridge PCI devices to a processor that implements the PowerPC architecture, and to serve as a PCI interface for CompactPCI™ (CPCI) systems or as a basis for passive PCI NIC implementations. In addition, multiple MPC8272 processors can interface with each other over the PCI bus.

The key features of the PCI bridge are as follows:

- PCI Specification, revision 2.2 compliant and supports frequencies up to 66 MHz
- On-chip arbitration
- Supports PCI-to-60x-memory and 60x-memory-to-PCI streaming
- PCI host bridge or peripheral capabilities
- Includes 4 DMA channels for the following transfers:
  - PCI-to-60x to 60x-to-PCI
  - 60x-to-PCI to PCI-to-60x
  - PCI-to-60x to PCI-to-60x
  - 60x-to-PCI to 60x-to-PCI
- Includes all of the configuration registers required by the PCI standard as well as message and doorbell registers
- Supports the I_2O standard
- Hot-Swap friendly (supports the Hot Swap Specification as defined by PICMG 2.1 R1.0 August 3, 1998)
- Supports 66-MHz, 3.3-V specification
- Uses a buffer pool for the 60x-PCI bus interface



[1] $\overline{\text{PCI\_MODE}}$ should be grounded at all times.

Figure 9-1. PCI Bridge in the MPC8272

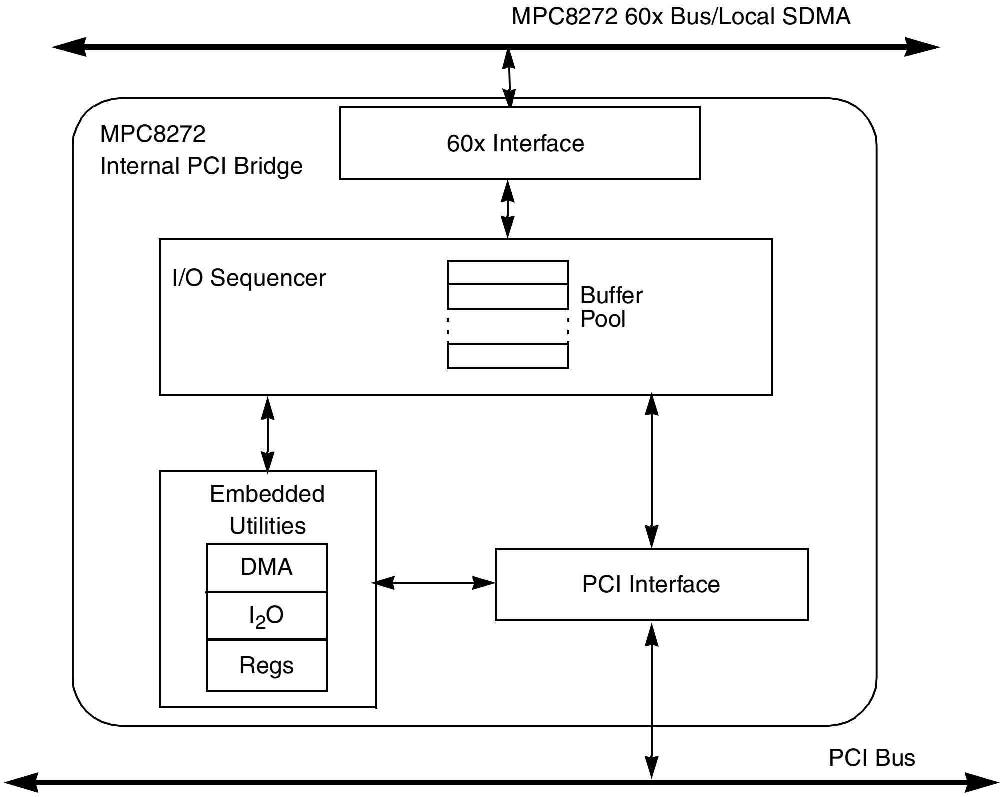


Figure 9-2. PCI Bridge Structure

9.1 Signals

PCI bridge signals are described in Chapter 6, “External Signals.”

9.2 Clocking

PCI bridge clocking is described in Chapter 10, “PLL and Clock Generator.”

9.3 PCI Bridge Initialization

The PCI bridge uses fields from the hard reset configuration word (refer to Section 5.4.1, “Hard Reset Configuration Word”) that are loaded during a hard reset (that is, assertion of the $\overline{\text{HRESET}}$ signal). This section discusses PCI bridge initialization issues after reset.

For PCI agent applications, the $\overline{\text{PCI\_RST}}$ signal should be connected to the power-on reset ($\overline{\text{PORESET}}$) pin of the MPC8272. If the core is disabled, in PCI agent mode, an EEPROM must be provided for loading the PCI configuration data.

For core-disabled, PCI agent applications, the communications processor (CP) can perform the minimal initialization of the internal PCI bridge configuration registers required before responding to configuration cycles. When the auto-load enable (ALD_EN) bit is set in the hard reset configuration word, the CP automatically loads the PCI configuration data from the EEPROM immediately following hard reset. (In addition to the hard reset configuration word, the PCI configuration register data should be programmed within the EEPROM according to the port size. Refer to configuration register loading in Section 9.11.2.28, “Initializing the PCI Configuration Registers,” for further details.) To prevent premature accesses, CFG_LOCK (see Section 9.11.2.22, “PCI Bus Function Register”) is automatically set during hard reset so that all attempted PCI accesses are retried. The user must re-enable PCI accesses by clearing CFG_LOCK at the end of the PCI bridge initialization procedure.

In addition to the configuration register programming, several configuration pins are available in PCI mode only. See Table 6-1 for a description of the external signals.

9.4 SDMA Interface

As shown in Figure 9-1, the PCI bridge has an interface to the SDMA controller. The CP can direct the SDMA controller to bring data from the PCI bus memory/IO space into the dual-port RAM, or vice versa. The user can choose if the data buffers, buffer descriptors, or any other needed data will reside on the 60x bus or on the PCI bus. $\overline{\text{PCI\_MODE}}$ should be grounded at all times.

NOTE

> Although the user can direct the SDMA to the 60x bus, transactions can be redirected to the PCI bridge if they fall in one of the PCI windows of the 60x bus memory map (PCIBR0 or PCIBR1; refer to Section 4.3.4.1, “PCI Base Register (PCIBRx)”). Data flow of this kind is not recommended because it is not optimal. However, if it is implemented, the user must set strict 60x bus mode (BCR[ETM] = 0).

9.5 Interrupts from PCI Bridge

Each of the PCI bridge interrupt sources—the PCI error condition detector, the DMA unit, and the message unit—can generate an interrupt to the SIU interrupt controller. PCI bridge interrupts are reflected in SIPNR_H[PCI] (refer to Section 4.3.1.4, “SIU Interrupt Pending Registers (SIPNR_H and SIPNR_L)”). PCI bridge interrupts can be masked in general with SIMR_H[PCI] (refer to Section 4.3.1.5, “SIU Interrupt Mask Registers (SIMR_H and SIMR_L)”). Specific interrupt sources can be masked independently by masking the relevant bits in the following registers—error mask register, DMA mode register, inbound message interrupt mask register, and the outbound message interrupt mask register. Each of these registers is described in Section 9.11.1, “Memory-Mapped Configuration Registers.”

The interrupt service routine can determine the source of the interrupt by reading the status bits of the following registers—the error status register, the DMA general status register, the inbound message interrupt status register, and the outbound message interrupt status register.

For PCI interrupt vector calculation, refer to Section 4.2.4, “Interrupt Vector Generation and Calculation.”

For the priority of PCI interrupts, refer to Section 4.3.1.2, “SIU Interrupt Priority Register (SIPRR).”

9.6 60x Bus Arbitration Priority

To prevent 60x bus arbitration deadlock, the PCI bridge should be programmed to have a high arbitration priority level within the 60x bus. The 60x bus arbitration-level register (PPC_ALRH) should be programmed so that the PCI request level index (0b0011) has a priority higher than all other 60x bus masters that address the PCI space through the 60x-PCI bridge (that is, the internal core or any external masters). Masters that do not perform transactions in the PCI space (through the 60x-PCI bridge) can have higher priority. Note that the default value of ALRH (0x0126_7893l) does not meet this requirement.

The same guidelines to prevent 60x bus arbitration deadlock apply when programming the parked master. That is, program the parked master in the 60x bus arbiter configuration register (PPC_ACR[PRKM]) to be the PCI bridge (0b0011); refer to Section 4.3.2.2, “60x Bus Arbiter Configuration Register (PPC_ACR).”

9.7 60x Bus Masters

The number of external 60x bus masters allowed access to the PCI bridge is limited by the number of pending requests that the PCI bridge is able to service. This number depends on the processor type of the master. For example, up to two second-generation (G2) processors that implement the PowerPC architecture or three third-generation (G3) processors can be accommodated.

9.8 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 three levels of support:

- Hot Swap capable
- Hot Swap friendly
- Hot Swap ready

The MPC8272 is a Hot Swap–friendly device, 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 MPC8272 device as a PCI target device. The only compliance exception is that the device pins are not 5-volt tolerant. Application boards should be used in 3.3-V signaling back planes, or add 5-to-3.3 volt signaling voltage converters, if needed, to be used in 5-V back planes.

For more information regarding the Hot Swap process, refer to the Hot Swap Specification PICMG 2.1, R1.0, August 3, 1998.

9.9 MPC8272 PCI Interface

The PCI bridge connects the processor and memory system to the I/O components through the PCI system bus. This interface acts as both an initiator (master) and target (slave) device. The PCI bridge 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—32-bit address memory, 32-bit address I/O, and PCI configuration space.

The PCI bridge can function as either a host bridge or an agent device. Note that the PCI bridge can be configured from the PCI bus while in agent mode. An address translation mechanism is provided to map PCI memory windows between the host and agent.

The following are the major features supported by the PCI interface:

- PCI Specification, revision 2.2 compliant
- On-chip arbitration supports three external PCI bus masters (in addition to the PCI bridge itself)
- Arbiter supports high-priority request and grant signal pairs
- Supports accesses to all PCI address spaces
- Supports PCI-to-60x-memory and 60x-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 snoop
- PCI host bridge capabilities
- PCI agent mode capabilities that include the ability to configure from a remote host
- Address translation units for address mapping between host and agent.

Efforts were made to keep the terminology in this chapter consistent with the PCI Specification, revision 2.2, and other PCI documentation; therefore, the terms found in Table 9-1 may differ from most documentation for processors that implement the PowerPC architecture (for example, architecture specifications or reference manuals).

Table 9-1. PCI Terminology

Term	Definition
LSB/Lower order	Represents bit 0 or the bits closest to the LSB
MSB/High order	Represents bit 31 or the bits closest to the MSB
Byte	Represents 8 bits of information
Word	Represents 16 bits or 2 bytes
Double word	Represents 32 bits or 2 words or 4 bytes
Quad word	Represents 64 bits or 2 double words, 4 words, or 8 bytes
Beat	Represents any valid data during a data transfer
Burst	Represents any 1 or more beat transfers
Edge/Clock edge	Represents the rising edge of the PCI clock
Cycle/Clock cycle	Represents the time period between clock edges
Asserted/Negated	Represents the globally visible **state** of the signal on the clock edge
Address phase	Represents the first clock cycle where $\overline{\text{FRAME}}$ is asserted
Data phase(s)	Represents the clock cycle(s) where $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted

NOTE: PCI Bridge Signal Naming

PCI bridge signals are defined in most cases with the prefix “PCI_” (for example, PCI_IRDY—see Figure 6-1). In this chapter, however, the prefix is not used. For descriptions of PCI bridge signals, refer to Chapter 6, “External Signals.”

9.9.1 PCI Interface Operation

The following sections discuss the operation of the PCI bus.

9.9.1.1 Bus Commands

PCI bus commands indicate the type of transaction occurring on the bus. These commands are encoded on PCI_C/BE[3:0] during the address phase of the transaction. PCI bus commands are described in Table 9-2.

Table 9-2. PCI Command Definitions

PCI_C/BE[3:0]	Command Type	Supported as		Definition
		Initiator	Target	
0b0000	Interrupt acknowledge	Yes	No	A read implicitly addressed to the system interrupt controller. The size of the vector to be returned is indicated on the byte enables after the address phase.
0b0001	Special cycle	Yes	No	Provides a simple message broadcast mechanism. See Section 9.9.1.4.6, “Special Cycle Command.”
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 9.9.1.4.4, “Host Mode Configuration Access” for more detail of configuration accesses. As a target, a configuration read is only accepted if the PCI bridge 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 9.9.1.4.4, “Host Mode Configuration Access”. As a target, a configuration write is only accepted if the PCI bridge is configured to be in agent mode.
0b1100	Memory read multiple	Yes	Yes	Causes a prefetch of the next cache line

Table 9-2. PCI Command Definitions (continued)

PCI_C/BE[3:0]	Command Type	Supported as		Definition
		Initiator	Target	
0b1101	Dual address cycle	No	No	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

9.9.1.2 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.

9.9.1.2.1 Basic Transfer Control

PCI data transfers are controlled with three fundamental signals:

- FRAME is driven by an initiator to indicate the beginning and end of a transaction.
- IRDY (initiator ready) is driven by an initiator, allowing it to force wait cycles.
- TRDY (target ready) is driven by a target, allowing it to force wait cycles.

The bus is idle when both FRAME and IRDY are negated. The first clock cycle in which 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 IRDY and TRDY are asserted. Once the PCI bridge, as an initiator, has asserted IRDY, it does not change IRDY or FRAME until the current data phase completes, regardless of the state of TRDY. Once the PCI bridge, as a target, has asserted TRDY or STOP, it does not change DEVSEL, TRDY, or STOP until the current data phase completes.

When the PCI bridge (as a master) intends to complete only one more data transfer, FRAME is negated and IRDY is asserted (or kept asserted) indicating the initiator is ready. After the target indicates it is ready (TRDY asserted) the bus returns to the idle state.

9.9.1.2.2 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 has been defined specifically to support PCI hardware configuration. Each PCI device decodes the address for each PCI transaction with each agent responsible for its own address decode.

The information contained in the 2 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 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' (refer to Section 9.9.1.3, “Bus Transactions”).

In the configuration address space, accesses are decoded to a double-word 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 bridge determines a configuration access is for a device on the PCI bus by decoding a configuration command. When in agent mode, the PCI bridge responds to host-generated PCI configuration cycles when its IDSEL is asserted during a configuration cycle.

For memory accesses, the double-word address is decoded using AD[31:2]; thereafter, the address is incremented internally by one double word (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 bridge 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 bridge linearly increments the burst order starting at the critical word, 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/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.

9.9.1.2.3 Byte Enable Signals

The byte enable signals (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 bridge, 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 bridge expects the data not to be changed, and on a write transaction, the data is not stored.

9.9.1.2.4 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. IRDY, TRDY, and DEVSEL use the address phase as their turnaround-cycle. FRAME, PCI_C/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 FRAME and 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.

9.9.1.3 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.

9.9.1.3.1 Read and Write Transactions

Both read and write transactions begin with an address phase followed by a data phase. The address phase occurs when $\overline{\text{FRAME}}$ 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 transaction.

A read transaction starts when $\overline{\text{FRAME}}$ is asserted for the first time and the PCI_C/$\overline{\text{BE}}$[3:0] signals indicate a read command. Figure 9-3 shows an example of a single-beat read transaction.

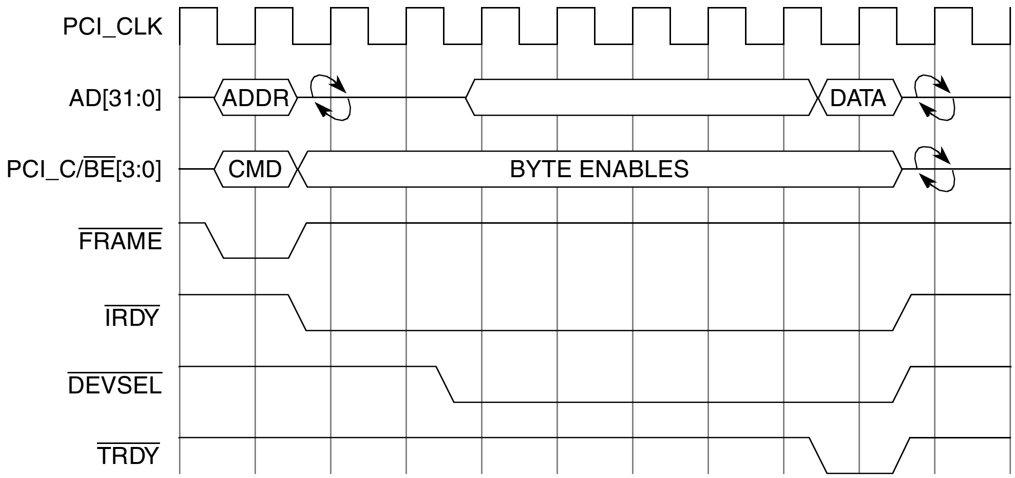


Figure 9-3. Single Beat Read Example

Figure 9-4 shows an example of a burst read transaction.

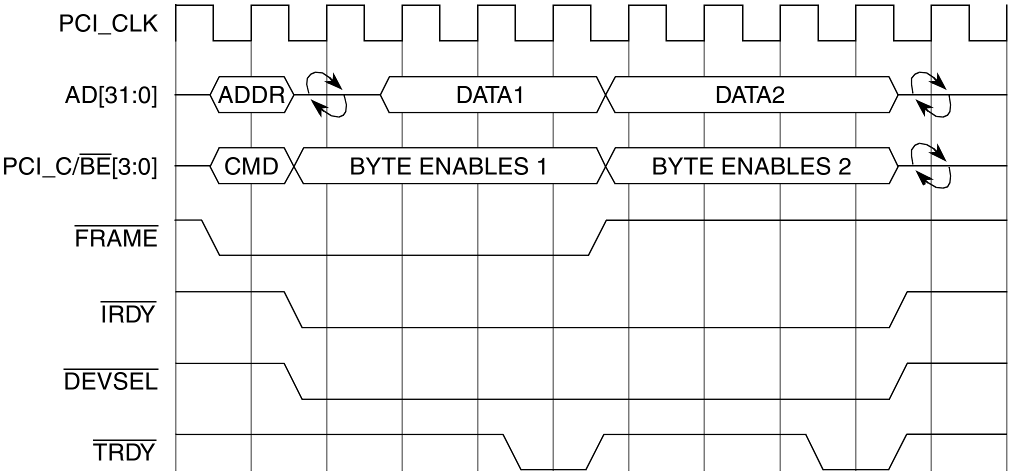


Figure 9-4. Burst Read Example

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 TRDY signal if using fast 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 DEVSEL is asserted.

The data phase completes when data is transferred, which occurs when both IRDY and 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 IRDY must be asserted when FRAME is negated.

A write transaction starts when FRAME is asserted for the first time and the PCI_C/BE[3:0] signals indicate a write command. Figure 9-5 shows an example of a single-beat write transaction.

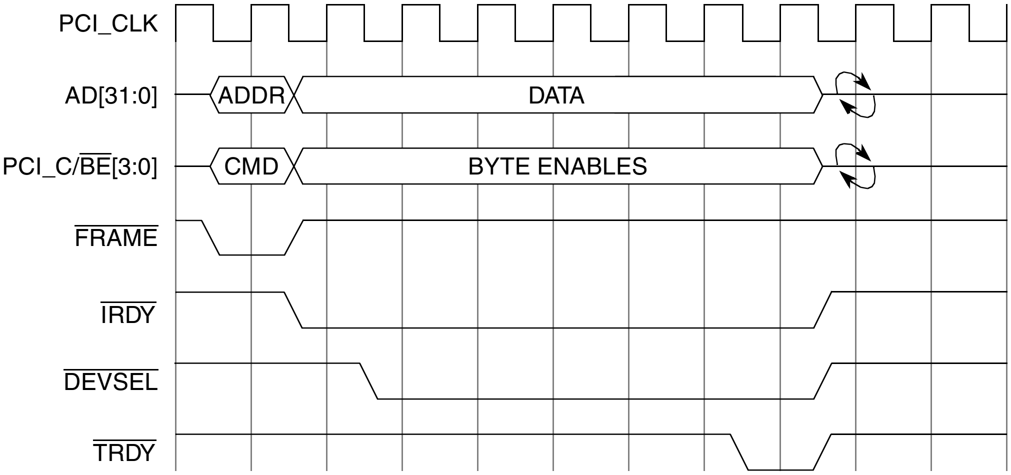


Figure 9-5. Single Beat Write Example

Figure 9-6 shows an example of a burst write transaction.

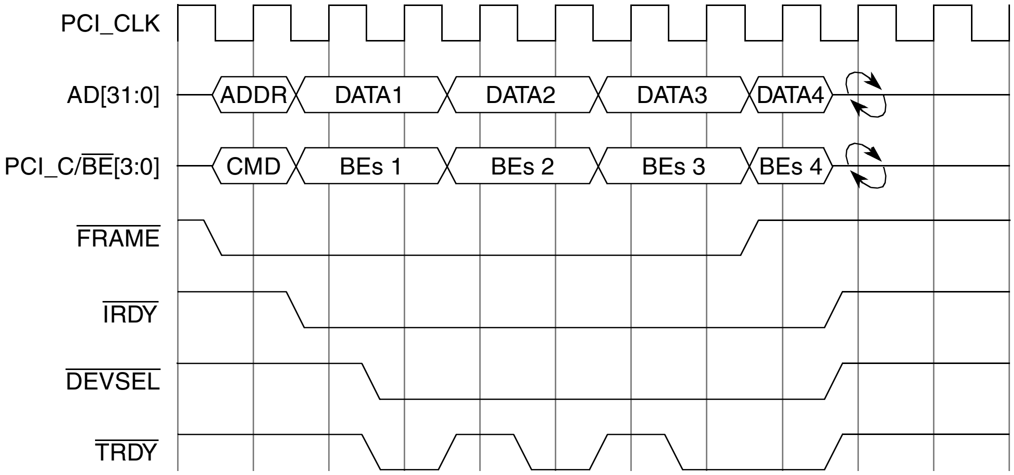

PCI_CLK
AD[31:0]
ADDR
DATA1
DATA2
DATA3
DATA4
PCI_C/$\overline{\mathrm{BE}}$[3:0]
CMD
BEs 1
BEs 2
BEs 3
BEs 4
$\overline{\mathrm{FRAME}}$
$\overline{\mathrm{IRDY}}$
$\overline{\mathrm{DEVSEL}}$
$\overline{\mathrm{TRDY}}$

Figure 9-6. 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.

9.9.1.3.2 Transaction Termination

The termination of a PCI transaction is orderly and systematic, regardless of the cause of the termination. All transactions end when $\overline{\mathrm{FRAME}}$ and $\overline{\mathrm{IRDY}}$ are both negated, indicating the idle cycle.

The PCI bridge as an initiator terminates a transaction when $\overline{\mathrm{FRAME}}$ is negated and $\overline{\mathrm{IRDY}}$ is asserted. This indicates that the final data phase is in progress. The final data transfer occurs when both $\overline{\mathrm{TRDY}}$ and $\overline{\mathrm{IRDY}}$ are asserted. A master-abort is an abnormal case of an initiated termination. If the PCI bridge detects that $\overline{\mathrm{DEVSEL}}$ has remained negated for more than four clocks after the assertion of $\overline{\mathrm{FRAME}}$, it negates $\overline{\mathrm{FRAME}}$ and, on the next clock, negates $\overline{\mathrm{IRDY}}$. On aborted reads, the PCI bridge returns 0xFFFF_FFFF. The data is lost on aborted writes.

When the PCI bridge as a target needs to suspend a transaction, it asserts $\overline{\mathrm{STOP}}$. Once asserted, $\overline{\mathrm{STOP}}$ remains asserted until $\overline{\mathrm{FRAME}}$ is negated. Depending on the circumstances, data may or may not be transferred during the request for termination. If $\overline{\mathrm{TRDY}}$ and $\overline{\mathrm{IRDY}}$ are asserted during the assertion of $\overline{\mathrm{STOP}}$**,** data is transferred. This type of target-initiated termination is called a ‘disconnect B,’ shown in Figure 9-7. If $\overline{\mathrm{TRDY}}$ is asserted when $\overline{\mathrm{STOP}}$ is asserted but $\overline{\mathrm{IRDY}}$ is not, $\overline{\mathrm{TRDY}}$ must remain asserted until $\overline{\mathrm{IRDY}}$ is asserted and the data is transferred. This is called a 'disconnect A' target-initiated termination, also shown in Figure 9-7. However, if $\overline{\mathrm{TRDY}}$ is negated when $\overline{\mathrm{STOP}}$ 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 9-7).

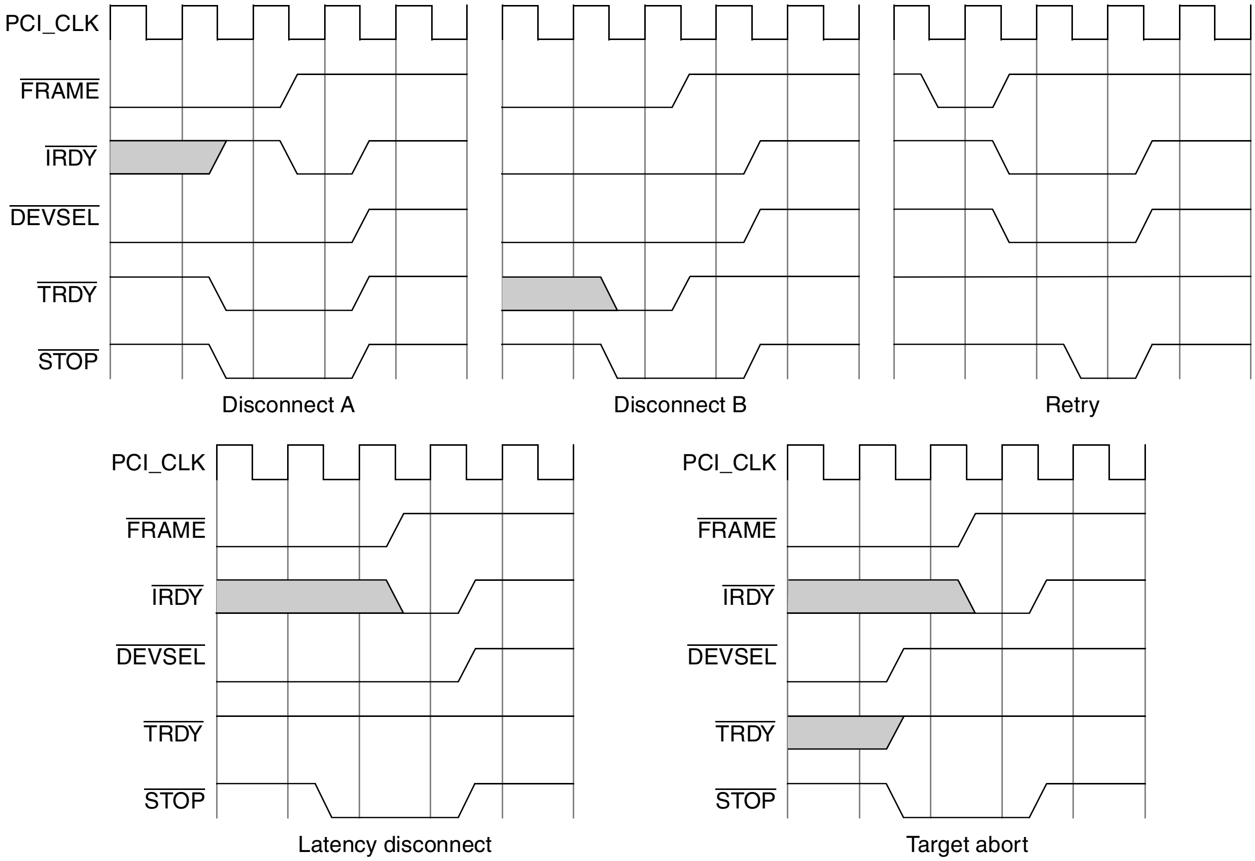


Figure 9-7. Target-Initiated Terminations

Note that when an initiator is terminated by $\overline{\text{STOP}}$, it must negate its $\overline{\text{REQ}x}$ 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 $\overline{\text{REQ}x}$ immediately following the two clocks or potential starvation may occur. If the initiator does not intend to complete the transaction, it can assert $\overline{\text{REQ}x}$ whenever it needs to use the PCI bus again.

The PCI bridge terminates a transaction in the following cases:

- Eight PCI clock cycles have elapsed between data phases. This is a ‘latency disconnect’ (see Figure 9-7).
- 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 when a streaming transaction crosses a 4K page boundary.
- A streaming transaction runs out of I/O sequencer buffer entries.
- A cache line wrap transaction has completed a cache line transfer.

Another target-initiated termination is the retry termination. Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. This can occur because no buffer entries are available in the I/O sequencer, or the sixteen clock latency timer has expired without transfer of the first data. The target latency timer of the PCI bridge can be optionally disabled; see Section 9.11.2.22, “PCI Bus Function Register.”

When the PCI bridge is in host mode it does not respond to any PCI configuration transactions. When the PCI bridge is in agent mode and AGENT_CFG_LOCK is set (refer to Section 9.11.2.22, “PCI Bus Function Register”) the PCI bridge retries all configuration transactions. Note that all retried accesses need to be completed. An example of a retry is shown in Figure 9-7.

Note that because a target can determine whether or not data is transferred (when both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted), if it wants to do only one more data transfer and then stop, it may assert $\overline{\text{TRDY}}$ and $\overline{\text{STOP}}$ 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 by the fact that $\overline{\text{STOP}}$ is asserted and $\overline{\text{DEVSEL}}$ 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 bridge 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 bridge 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 bridge does not target-abort in this case.

If the PCI bridge is mastering a transaction and the transaction terminates with a target-abort, undefined data will be returned on a read and write data will be lost. An example of a target-abort is shown in Figure 9-7.

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.

9.9.1.4 Other Bus Operations

The following sections provide information on additional PCI bus operations.

9.9.1.4.1 Device Selection

As a target, the PCI bridge drives $\overline{\text{DEVSEL}}$ one clock following the address phase as indicated in the configuration space status register; see Section 9.11.2.4, “PCI Bus Status Register.” The PCI bridge as a target qualifies the address/data lines with $\overline{\text{FRAME}}$ before asserting $\overline{\text{DEVSEL}}$. $\overline{\text{DEVSEL}}$ is asserted at or before the clock edge at which the PCI bridge enables its $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, or data (for a read). $\overline{\text{DEVSEL}}$ is not negated until $\overline{\text{FRAME}}$ is negated, with $\overline{\text{IRDY}}$ asserted and either $\overline{\text{STOP}}$ or $\overline{\text{TRDY}}$ asserted. The exception to this is a target-abort; see Section 9.9.1.3.2, “Transaction Termination.”

As an initiator, if the PCI bridge does not see the assertion of $\overline{\text{DEVSEL}}$ within four clocks of $\overline{\text{FRAME}}$, it terminates the transaction with a master-abort as described in Section 9.9.1.3.2, “Transaction Termination.”

9.9.1.4.2 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 bridge as a target supports both types of fast back-to-back transactions but does not support them as an initiator. The PCI bridge as a target has the fast back-to-back enable bit hardwired to one, or enabled; see Table 9-18.

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 bridge detects a fast-back-to-back operation and did not drive $\overline{\text{DEVSEL}}$ in the previous cycle, it delays the assertion of $\overline{\text{DEVSEL}}$ and $\overline{\text{TRDY}}$ for one cycle to allow the other target to get off the bus.

9.9.1.4.3 Data Streaming

The PCI bridge provides data streaming for PCI transactions to and from prefetchable memory. In other words, when the PCI bridge 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 bridge disconnects after the first data phase so that no streaming can occur.

For PCI memory reads, streaming is achieved by performing speculative reads from memory in prefetchable space. A block of memory can be marked as prefetchable by setting the prefetch bit in the corresponding inbound ATU (see Table 9-18) 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 bridge reads one cache line from memory. If the PCI read or read line transaction crosses a cache line boundary, the PCI bridge starts the read of a new cache line. For a memory read multiple command, the PCI bridge reads two cache lines from memory. When the PCI transaction finishes the read for the first cache line, the PCI bridge performs a speculative read of a third cache line. The PCI bridge 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 bridge runs out of buffer space on writes, or the PCI bridge cannot supply consecutive data beats for reads within eight PCI bus clocks of each other. A disconnect also occurs if the transaction crosses a 4K page boundary.

For core- or DMA-initiated transfers, the PCI bridge streams over cache line boundaries if the prefetch bit in the corresponding outbound ATU is enabled and the address space identified by the outbound ATU is marked as PCI memory space.

9.9.1.4.4 Host Mode Configuration Access

The PCI bridge provides two types of configuration accesses to support hierarchical bridges. To access configuration space, a value is written to the CONFIG_ADDR register specifying the PCI bus, device, and configuration register to be accessed.

When the PCI bridge sees an access that falls inside the double-word 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 9.9.1.4.6, “Special Cycle Command”).

The format of CONFIG_ADDR is shown in Figure 9-8. Bits 23–16 choose a specific PCI bus in the system. Bits 15–11 choose a specific device on the bus. Bits 10–8 choose a specific function in the requested device. Bits 7–2 choose a DWORD in the device’s configuration space. Bit 31 is an enable flag for determining when accesses to CONFIG_DATA should be translated to configuration cycles.

31	30 24	23 16	15 11	10 8	7 2	1	0
E	—	Bus Number	Device Number	Function Number	Register Number	0	0

31 11	10 2	1	0
Only one bit is set at a time (for IDSEL)	Function Register	0	0

Figure 9-8. PCI Configuration Type 0 Translation (Top = CONFIG_ADDR) (Bottom = PCI Address Lines)

Two types of translations are supported:

- Type 0 translations—For when the device is on the PCI bus connected to the PCI bridge. (Figure 9-8 shows the type 0 translation from the CONFIG_ADDR register to the address/data lines on the PCI bus.)
- Type 1 translations—For when the device is on another bus somewhere behind the PCI bridge

For type 0 translations, the PCI bridge 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 AD[11]. That is, if the device number field contains 0b01011, AD[11] on the PCI bus is set. The IDSEL lines are bit-wise associated with increasing values for the device number such that AD[12] corresponds to 0b01100, and so on up to bit 30. AD[31] is selected with 0b01010. A device number of 0b11111 indicates a special cycle. Device number 0b00000 is used for configuring the PCI bridge itself. Bits 10–8 are copied to the PCI bus as an encoded value for components that contain multiple functions. Bits 7–2 are also copied onto the PCI bus. The PCI bridge implements address stepping on configuration cycles so that the target’s 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 $\overline{\text{FRAME}}$.

For type 1 translations, the PCI bridge 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).

NOTE

Due to design constraints, the software must write a value to the CONFIG_ADDR register prior to each access to the CONFIG_DATA register, even if the address was not changed.

When the MPC8272 is configured as a host device, it 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 'PCI No response' bit in the error mask register (clear bit 3). Refer to Section 9.11.1.9, “Error Status Register (ESR).”
2. Make the PCI configuration reads.
3. Clear bit 3 in the error status register (by writing 0x08).
4. Unmask (write '1') bit 3 in the error mask register. Refer to Section 9.11.1.10, “Error Mask Register (EMR).”

9.9.1.4.5 Agent Mode Configuration Access

When the PCI bridge 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 bridge's IDSEL being asserted. A remote host can access the 256-byte PCI configuration area (Figure 9-32) and the memory-mapped configuration registers within the PCI bridge.

9.9.1.4.6 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 $\overline{\text{DEVSEL}}$ 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 $\overline{\text{FRAME}}$ and completes when $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ 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 16 least-significant bits (AD[15:0]). The data field is encoded on AD[31:16]. When running a special cycle, the PCI bridge can insert wait states, but because no specific target is addressed, the message and data are valid on the first clock $\overline{\text{IRDY}}$ is asserted.

When the CONFIG_ADDRESS register gets written with a value such that the bus number matches the bridge’s bus, the device number is all ones, the function number is all ones and the register number is zero, the next time the CONFIG_DATA register is accessed the PCI bridge does either a special cycle or an interrupt acknowledge command. When the CONFIG_DATA register is written, the PCI bridge generates a special cycle encoding on the command/byte enable lines during the address phase, and drives the data from the CONFIG_DATA register onto the address/data lines during the first data phase.

If the bus number field of the CONFIG_ADDRESS does not match one of the PCI bridge’s bus numbers, the PCI bridge passes the write to CONFIG_DATA on through to the PCI bus as a type 1 configuration cycle like any other time the bus number field does not match.

9.9.1.4.7 Interrupt Acknowledge

When the CONFIG_ADDRESS register gets written with a value such that the bus number is 0x00, the device number is all ones, the function number is all ones and the register number is zero, the next time the CONFIG_DATA register is accessed the PCI bridge does either a special cycle command or an interrupt acknowledge command. When the CONFIG_DATA register is read, the PCI bridge 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 (PAR). During the data phase, the byte enable signals determine which bytes are involved in the transaction. The interrupt vector must be returned when TRDY is asserted.

An interrupt acknowledge transaction can also be issued on the PCI bus by reading from the PCI_INT_ACK register.

9.9.1.5 Error Functions

This section discusses PCI bus errors.

9.9.1.5.1 Parity

During valid 32-bit address and data transfers, parity covers all 32 address/data lines and the four 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 PAR is generated such that the number of ones on AD[31:0]**,** PCI_C/BE[3:0], and PAR equals an even number. PAR is driven when the address/data lines are driven and follow the corresponding address or data by one clock.

The PCI bridge checks the parity after all valid address phases (the assertion of FRAME) and for valid data transfers (IRDY and TRDY asserted) involving the PCI bridge. When an address or data parity error is detected, the detected-parity-error bit in the configuration space status register is set (see Section 9.11.2.4, “PCI Bus Status Register”).

9.9.1.5.2 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 9.11.2.3, “PCI Bus Command Register”). If the parity-error-response bit is cleared, the PCI bridge completes all transactions regardless of parity errors (address or data). If the bit is set, the PCI bridge asserts PERR two clocks after the actual data transfer in which a data parity error is detected, and keeps PERR asserted for one clock. The PCI bridge asserts PERR when acting as an initiator during a read transaction or as a target involved in a write to system memory. Figure 9-9 shows the possible assertion points for PERR if the PCI bridge detects a data parity error.

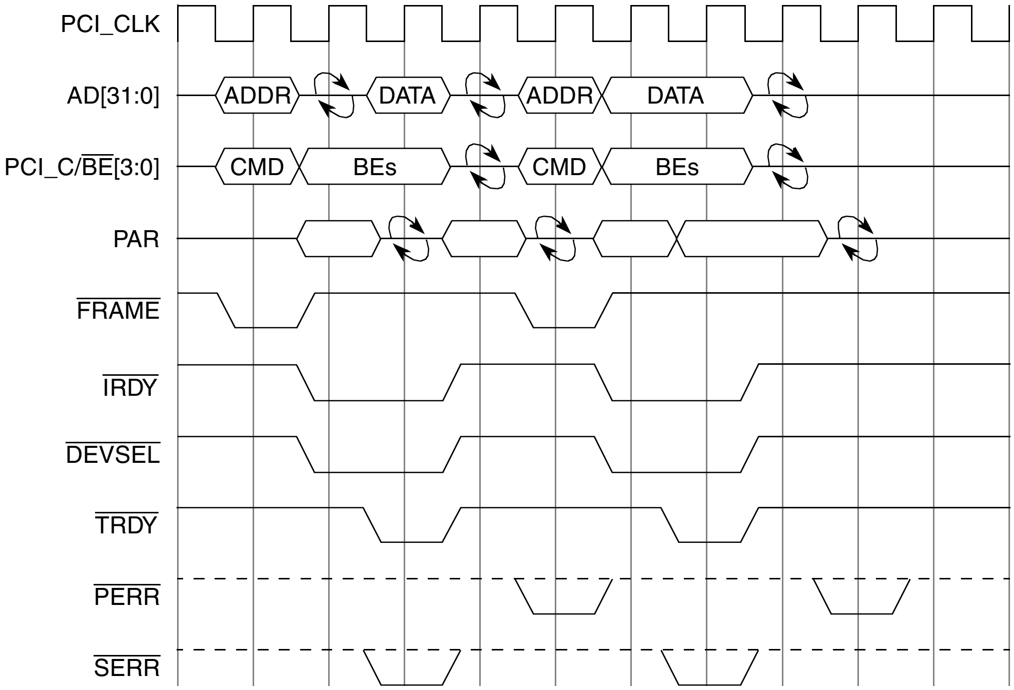


Figure 9-9. PCI Parity Operation

As an initiator, the PCI bridge 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 bridge aborts the transaction internally. As a target, the PCI bridge 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, ensuring that potentially corrupt data does not go to memory.

When the PCI bridge asserts 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 SERR is conditioned on the parity-error-response bit being enabled in the command register. SERR is asserted when the PCI bridge detects an address parity error while acting as a target. The system error is passed to the PCI bridge's interrupt processing logic to assert MCP. Figure 9-9 shows where the PCI bridge could detect an address parity error and assert SERR or where the PCI bridge, acting as an initiator, checks for the assertion of SERR signaled by the target detecting an address parity error.

As a target that asserts SERR on an address parity, the PCI bridge completes the transaction on the PCI bus, aborting internally if the transaction is a write to system memory. If PERR is asserted during a PCI bridge write to PCI, the PCI bridge attempts to continue the transfer, allowing the target to abort/disconnect if desired. If the PCI bridge detects a parity error on a read from PCI, the PCI bridge 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; MCP is also asserted to the core as an option.

9.9.2 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 ($\overline{\mathrm{REQ}x}$) output and grant ($\overline{\mathrm{GNT}x}$) 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 bridge provides arbitration for three external PCI bus masters (besides the PCI bridge itself) by using the $\overline{\mathrm{REQ0}}$, $\overline{\mathrm{REQ1}}$, and $\overline{\mathrm{REQ2}}$ signals and generating the $\overline{\mathrm{GNT0}}$, $\overline{\mathrm{GNT1}}$, and $\overline{\mathrm{GNT2}}$ signals.

During reset, the PCI bridge samples the PCI_CFG[1] pin (and programs the PCI_ARB_DIS bit accordingly) to determine if the arbiter is enabled or disabled. The arbiter can also be enabled or disabled by directly programming the PCI_ARB_DIS bit in the arbiter configuration register (see Section 9.11.2.23, “PCI Bus Arbiter Configuration Register”).

If the arbiter is disabled, the PCI bridge uses $\overline{\mathrm{REQ0}}$ to issue requests to an external arbiter, and uses $\overline{\mathrm{GNT0}}$ to receive grants from the external arbiter.

The PCI bridge implements a two-level priority, round-robin arbitration algorithm. The priority level for the different masters can be programmed in the arbiter configuration register (see Section 9.11.2.23, “PCI Bus Arbiter Configuration Register”).

9.9.2.1 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/$\overline{\mathrm{BE}}$ and PAR signals from floating. The PCI bridge can be configured to either park on the PCI bridge or park on the last master to use the bus by programming the parking-mode bit in the arbiter configuration register (see Section 9.11.2.23, “PCI Bus Arbiter Configuration Register”).

9.9.2.2 Arbitration Algorithm

The arbitration algorithm implemented is round-robin with two priority levels. Each of the three external PCI bus masters, plus the PCI bridge, are assigned either a high or a low priority level, as programmed in the arbiter configuration register (see Section 9.11.2.23, “PCI Bus Arbiter Configuration Register”). Within each priority group (high or low), the bus grant is given to the next requesting device in numerical order, with the PCI bridge itself positioned before device 0. $\overline{\mathrm{GNT}x}$ is asserted for device *x* as soon as the previously granted device begins a transaction. Conceptually, the lowest priority device at any given time is the master that is currently using the bus, 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 a particular device may be taken away and given to another, higher priority device whenever the 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 and then 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 9-10. Noting that one position in the high priority group is actually a placeholder 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 bridge, 0, 2, 1, 0, 2, the PCI bridge, and so on. If, for example, device 2 is not requesting the bus, the grant sequence becomes 0, the PCI bridge, 0, 1, 0, the PCI bridge, 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 bridge is the next grant, then the PCI bridge's grant is removed, and the higher-priority device 2 is awarded the next grant.

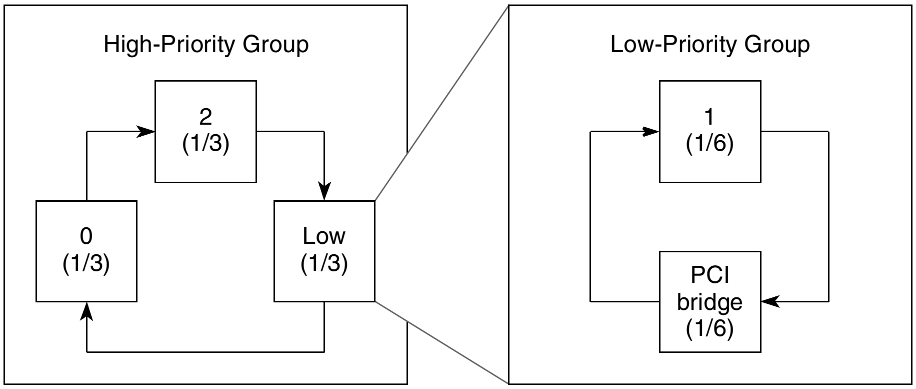


Figure 9-10. PCI Arbitration Example

9.9.2.3 Master Latency Timer

The PCI bridge implements the master latency timer register (see Section 9.11.2.10, "PCI Bus Latency Timer Register") to prevent the itself from monopolizing the bus. When the master latency timer expires, the PCI bridge checks the state of its $\overline{\text{GNT}}$ signals. If the $\overline{\text{GNT}}$ signal is not asserted, the PCI bridge completes one more data phase and relinquishes the bus. The master latency timer can be disabled if needed (see Section 9.11.2.22, "PCI Bus Function Register").

9.10 Address Map

A transaction sent to the PCI bridge from any 60x bus master side falls into one of the following three cases:

- If the transaction address is within the internal register space of the MPC8272, the transaction is handled by the PCI bridge internal register logic. (The internal registers are described in this chapter.)
- If the transaction address is within one of the three outbound PCI translation windows (described in this chapter), the transaction is sent to the PCI bus with address translation.
- If the transaction address is not within the internal register space and not within a PCI translation window, the transaction is sent to the PCI bus with no address translation as a PCI memory transaction to non-prefetchable space.

An address decode flow chart for transactions from the 60x bus masters to the PCI bridge is shown in Figure 9-11.

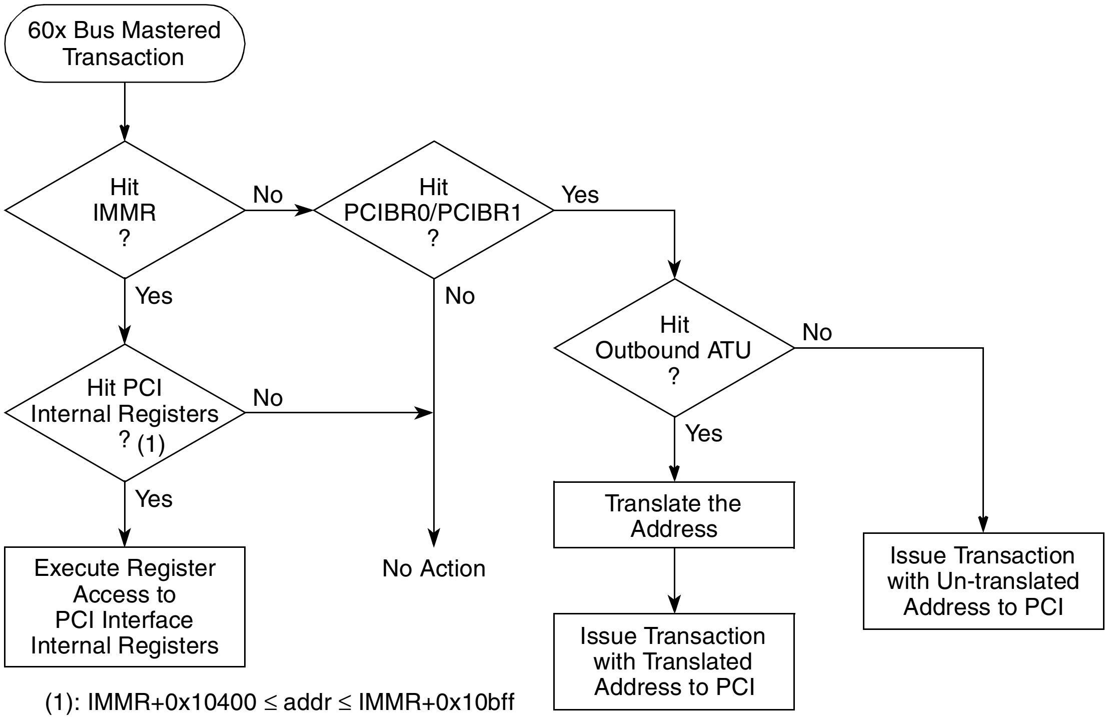

60x Bus Mastered Transaction
Hit IMMR ?
No
Hit PCIBR0/PCIBR1 ?
Yes
Yes
No
Hit Outbound ATU ?
No
Hit PCI Internal Registers ? (1)
No
Yes
Yes
Translate the Address
Execute Register Access to PCI Interface Internal Registers
No Action
Issue Transaction with Un-translated Address to PCI
Issue Transaction with Translated Address to PCI
(1): IMMR+0x10400 ≤ addr ≤ IMMR+0x10bff

Figure 9-11. Address Decode Flow Chart for 60x Bus Mastered Transactions

Transactions directed to the MPC8272 from a PCI bus master are handled as follows:

- If the transaction address is within the internal register space of the MPC8272, the transaction is either handled by the PCI bridge internal register logic or forwarded to the core side of the PCI bridge to be handled by the MPC8272 internal register logic as appropriate.
- If the transaction address is within one of the two inbound PCI translation windows, the transaction is sent to the core side of the PCI bridge with address translation.

NOTE

This window is provided for the PCI master to access the MPC8272's internal (dual port) registers/area. Its size is assumed to be fixed at 128 Kbytes. It translates to the IMMR value for the upper bits of the address. This way, the PCI master can access any of the PCI bridge registers (DMA/MU, and so on) without wasting an inbound translation window. In effect, it suggests that there is a total of three inbound windows, two with ATUs and one with PIMMR.

An address decode flow chart for transactions from a PCI bus master to the PCI bridge is shown in Figure 9-12.

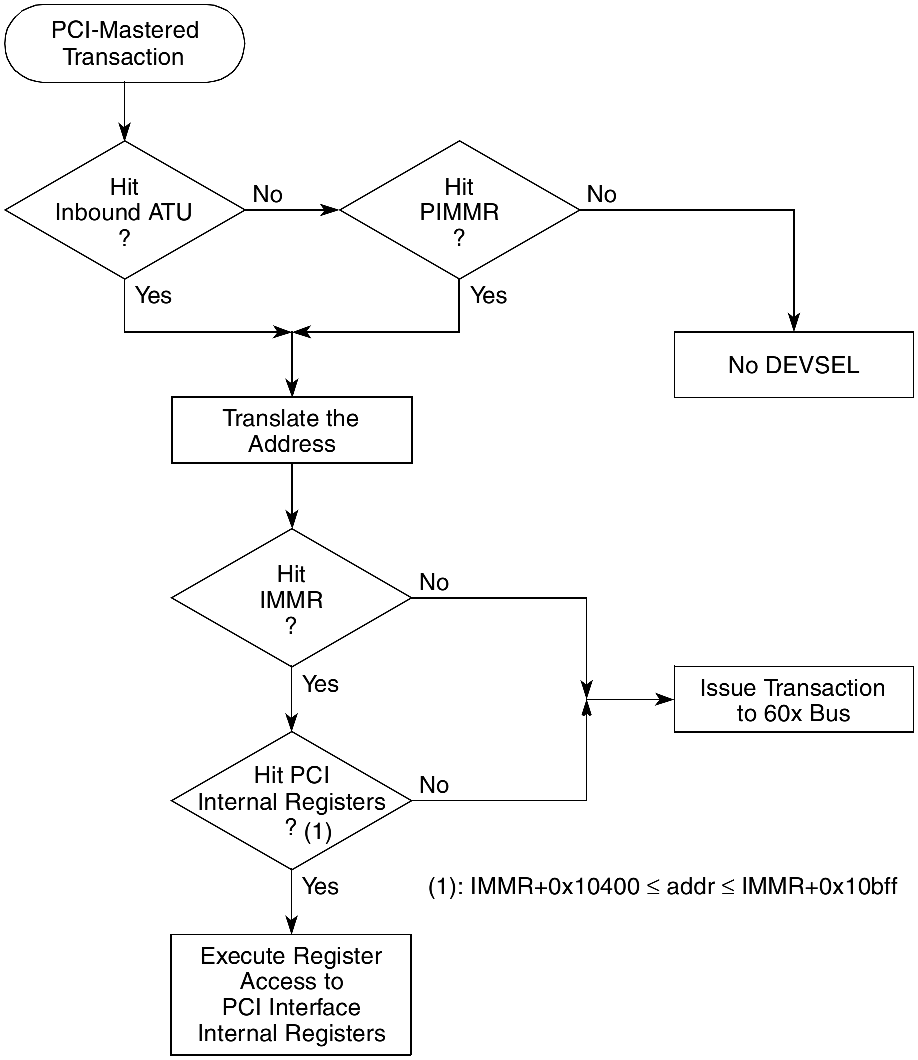

PCI-Mastered Transaction
Hit Inbound ATU ?
No
Hit PIMMR ?
No
Yes
Yes
No DEVSEL
Translate the Address
Hit IMMR ?
No
Yes
Issue Transaction to 60x Bus
Hit PCI Internal Registers ? (1)
No
Yes
(1): IMMR+0x10400 ≤ addr ≤ IMMR+0x10bff
Execute Register Access to PCI Interface Internal Registers

Figure 9-12. Address Decode Flow Chart for PCI Mastered Transactions

NOTE

When a transaction is performed by a PCI master, the bridge checks the address against inbound ATUs and, if it does not hit, checks against PIMMR; if it is a hit, the bridge translates it to a 60x cycle. Because PIMMR does not have an associated translation register and window size definition, the translation is performed as follows: a 128-Kbyte window is provided for the PCI master to access the MPC8272's internal (dual port) registers. It translates to the MPC8272's IMMR value for the upper bits of the address. This allows the PCI master to access any of the PCI-bridge registers without wasting an inbound translation window. In effect, there are a total of three inbound windows, two with ATUs and one with PIMMR.

Transactions initiated by the DMA controller or message unit fall into one of the following cases:

- If the transaction address is within one of the outbound PCI translation windows, the transaction is sent to the PCI bus with address translation.
- If the transaction address is not within a PCI translation window, the transaction is sent to the core side of the PCI bridge with no address translation.

An address decode flow chart for transactions from the DMA controller or message unit to the PCI bridge is shown in Figure 9-13.

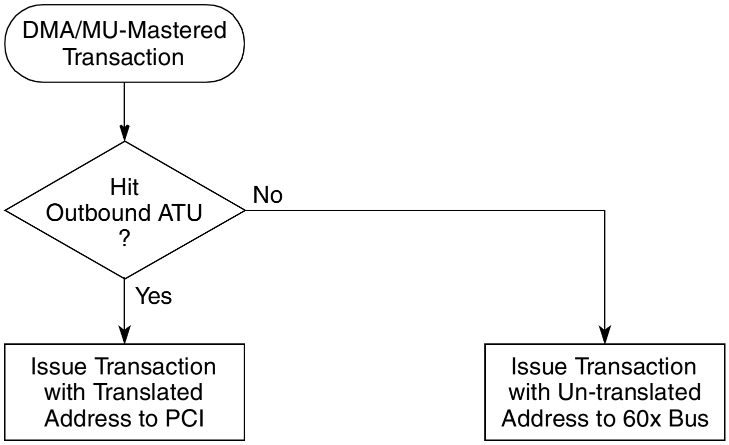

DMA/MU-Mastered Transaction
Hit Outbound ATU ?
No
Yes
Issue Transaction with Translated Address to PCI
Issue Transaction with Un-translated Address to 60x Bus

Figure 9-13. Address Decode Flow Chart for Embedded Utilities (DMA, Message Unit) Mastered Transactions

Example address mappings of these different types of transactions are shown in Figure 9-14. Note that the translation mechanism shown is an example only; the address translation, as well as the memory and I/O destinations, can be programmed independently for each address translation window.

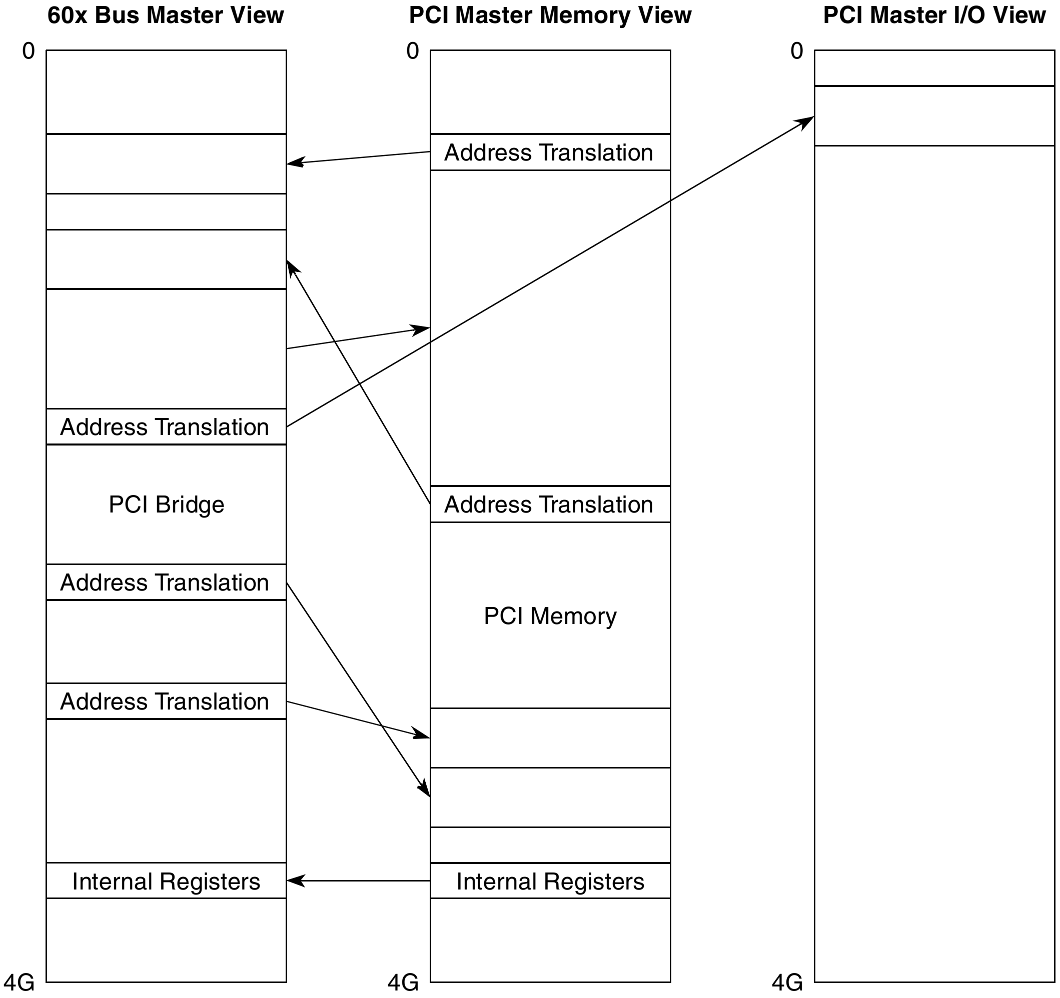


Figure 9-14. Address Map Example

9.10.1 Address Map Programming

The address map has a number of programmable ranges to determine the PCI bridge’s response to all transactions. The following are the rules for programming each address range:

- All address regions should not overlap but do not have to be contiguous.
- All address ranges must be aligned on a multiple of the region size.
- Inbound and outbound windows for the same bus should not overlap. This means that a situation where an inbound window translation points back into an outbound window, or a situation where an outbound translation window points back into an inbound window, are not allowed.

9.10.2 Address Translation

The address translation registers allow the remapping of inbound and outbound transactions. The reset configuration for outbound transactions are that all outbound requests from the core side of the PCI bridge are routed to the PCI bus with address translation disabled. The reset configuration for inbound transactions are that all inbound requests from the PCI bus are disabled.

9.10.2.1 PCI Inbound Translation

For inbound transactions (transactions generated by an external master on the PCI bus where the PCI bridge responds as a slave device), the PCI bridge only responds to PCI addresses within the windows mapped by the PCI inbound base address registers (PIBARs). If an address hit occurs 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 on the 60x's bus. Each PIBAR register is associated with a PITAR and PICMR (PCI inbound comparison mask register) which are located in the PCI bridge's PCI internal register space. Figure 9-15 shows an example translation window for inbound memory accesses.

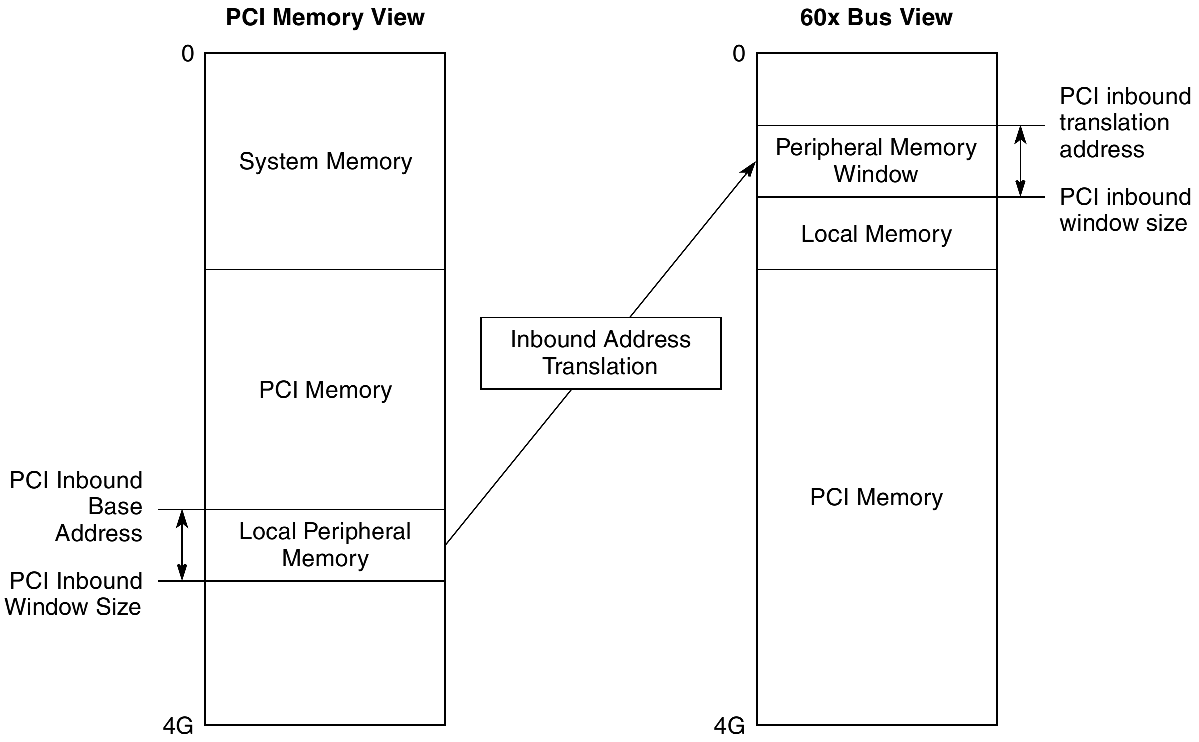


Figure 9-15. Inbound PCI Memory Address Translation

There are two sets of inbound translation registers, allowing two simultaneous translation windows. Software can move the translation base addresses during run-time to access different portions of local memory, but be sure that the PCI inbound translation windows do not overlap.

The reset configuration for the windows is disabled; that is, after reset, the PCI bridge does not acknowledge externally mastered transactions on the PCI bus by asserting $\overline{\text{DEVSEL}}$ until the inbound translation windows are enabled. The inbound translation is performed in the PCI interface.

9.10.2.2 PCI Outbound 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's base addresses are defined in the PCI outbound base address registers (refer to Section 9.11.1.4, "PCI Outbound Base Address Registers (POBARx)"). 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). Outbound addresses that fall outside the outbound windows are forwarded to the PCI bus without modification. Figure 9-16 shows an example translation window for outbound memory accesses.

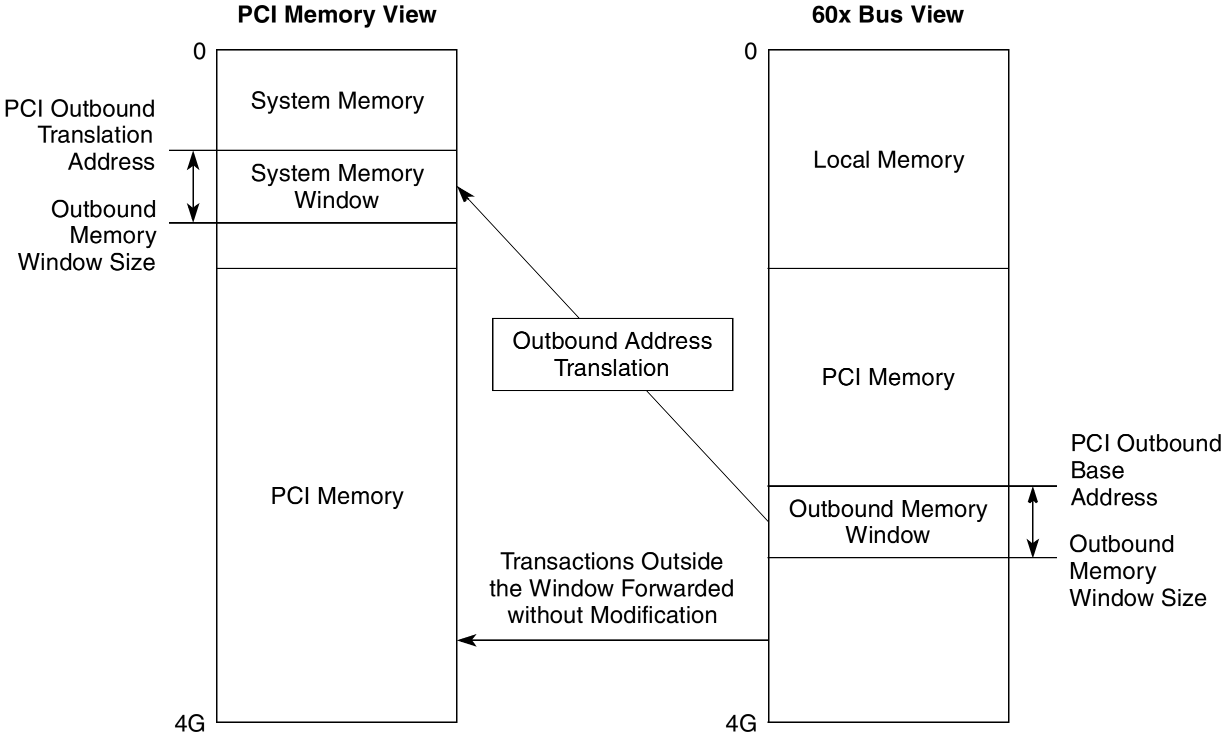

PCI Memory View
60x Bus View
0
0
PCI Outbound Translation Address
Outbound Memory Window Size
System Memory
System Memory Window
PCI Memory
Outbound Address Translation
Local Memory
PCI Memory
Outbound Memory Window
PCI Outbound Base Address
Outbound Memory Window Size
Transactions Outside the Window Forwarded without Modification
4G
4G

Figure 9-16. Outbound PCI Memory Address Translation

The three sets of outbound translation registers allow three simultaneous translation windows. Software can move and adjust the host memory window translations and sizes during run-time. This allows software to access host memory or to address alternate memory space on the fly, but be sure that the PCI outbound translation windows do not overlap. Also note that the PCI outbound translation windows should not overlap with the PCI bridge internal register space defined by the PIMMR.

9.10.3 SIU Registers

PCI utilizes fields in general SIU registers (SIUMCR, TESCR1, TESCR2, and L-TESCR1). There are also two pairs of PCI-specific registers that detect accesses from the 60x bus side to the PCI bridge (other than PCI internal registers accesses). Refer to Section 4.3.4, "PCI Control Registers."

9.11 Configuration Registers

There are two types of configuration registers in the PCI bridge: PCI-specified and memory-mapped. The PCI-specified type, referred to as PCI configuration registers, are accessed through PCI configuration cycles (refer to Section 9.11.2, "PCI Bridge Configuration Registers"). The memory-mapped configuration registers are placed in the internal memory map of the MPC8272 and are accessed like other internal registers (refer to Section 9.11.1, "Memory-Mapped Configuration Registers").

Both the PCI configuration and memory-mapped internal registers of the PCI bridge are intrinsically little endian and are described using classic bit numbering; that is, the lowest memory address contains the least-significant byte of the register and bit 0 is the least-significant bit of the register.

NOTE: Accessing Configuration Registers

For a PCI device to share little-endian (LE) data with the 603e core CPU, software must byte-swap the data of the configuration register. Refer to Section 9.11.2.27, "PCI Configuration Register Access in Big-Endian Mode," and Section 9.11.2.27.1, "Additional Information on Endianness."

Also note that reserved bits in the configuration registers are not guaranteed to have predictable values. Software must preserve the values of reserved bits when writing to a configuration register. Also, when reading from a configuration register, software should not rely on the value of reserved bits remaining constant.

NOTE: Accessing PCI Registers in Non-PCI Mode

In non-PCI mode, a 60x bus master should not attempt to access the PCI memory mapped configuration registers. Doing so will cause the internal memory space of the MPC8272 to be inaccessible. Any following access to the internal memory space will not be terminated normally, and can only be terminated by TEA if the 60x bus monitor is activated. The system can recover only after a soft reset.

9.11.1 Memory-Mapped Configuration Registers

Table 9-3 describes the memory-mapped configuration registers provided by the PCI bridge. Note that memory gaps not defined are reserved and should not be accessed.

Table 9-3. Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x10430	Outbound interrupt status register (OMISR)	special	0x0000_0000	9.12.3.4.3/9-80
0x10434	Outbound interrupt mask register (OMIMR)	R/W	0x0000_0000	9.12.3.4.4/9-81
0x10440	Inbound FIFO queue port register (IFQPR)	R/W	0x0000_0000	9.12.3.4.1/9-79
0x10444	Outbound FIFO queue port register (OFQPR)	R/W	0x0000_0000	9.12.3.4.2/9-80
0x10450	Inbound message register 0 (IMR0)	R/W	Undefined	9.12.1.1/9-68
0x10454	Inbound message register 1 (IMR1)	R/W	Undefined	9.12.1.1/9-68

Table 9-3. Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x10458	Outbound message register 0 (OMR0)	R/W	Undefined	9.12.1.2/9-69
0x1045C	Outbound message register 1 (OMR1)	R/W	Undefined	9.12.1.2/9-69
0x10460	Outbound doorbell register (ODR)	R/W	0x0000_0000	9.12.2.1/9-70
0x10468	Inbound doorbell register (IDR)	R/W	0x0000_0000	9.12.2.2/9-70
0x10480	Inbound message interrupt status register (IMISR)	R/W	0x0000_0000	9.12.3.4.5/9-82
0x10484	Inbound message interrupt mask register (IMIMR)	R/W	0x0000_0000	9.12.3.4.6/9-84
0x104A0	Inbound free_FIFO head pointer register (IFHPR)	R/W	0x0000_0000	9.12.3.2.1/9-73
0x104A8	Inbound free_FIFO tail pointer register (IFTPR)	R/W	0x0000_0000	9.12.3.2.1/9-73
0x104B0	Inbound post_FIFO head pointer register (IPHPR)	R/W	0x0000_0000	9.12.3.2.2/9-74
0x104B8	Inbound post_FIFO tail pointer register (IPTPR)	R/W	0x0000_0000	9.12.3.2.2/9-74
0x104C0	Outbound free_FIFO head pointer register (OFHPR)	R/W	0x0000_0000	9.12.3.3.1/9-76
0x104C8	Outbound free_FIFO tail pointer register (OFTPR)	R/W	0x0000_0000	9.12.3.3.1/9-76
0x104D0	Outbound post_FIFO head pointer register (OPHPR)	R/W	0x0000_0000	9.12.3.3.2/9-77
0x104D8	Outbound post_FIFO tail pointer register (OPTPR)	R/W	0x0000_0000	9.12.3.3.2/9-77
0x104E4	Message unit control register (MUCR)	R/W	0x0000_0002	9.12.3.4.7/9-85
0x104F0	Queue base address register (QBAR)	R/W	0x0000_0000	9.12.3.4.8/9-86
0x10500	DMA 0 mode register (DMAMR0)	R/W	0x0000_0000	9.13.1.6.1/9-90
0x10504	DMA 0 status register (DMASR0)	R/W	0x0000_0000	9.13.1.6.2/9-92
0x10508	DMA 0 current descriptor address register (DMACDAR0)	R/W	0x0000_0000	9.13.1.6.3/9-93
0x10510	DMA 0 source address register (DMASAR0)	R/W	0x0000_0000	9.13.1.6.4/9-94
0x10518	DMA 0 destination address register (DMADAR0)	R/W	0x0000_0000	9.13.1.6.5/9-95
0x10520	DMA 0 byte count register (DMABCR0)	R/W	0x0000_0000	9.13.1.6.6/9-95
0x10524	DMA 0 next descriptor address register (DMANDAR0)	R/W	0x0000_0000	9.13.1.6.7/9-96
0x10580	DMA 1 mode register (DMAMR1)	R/W	0x0000_0000	9.13.1.6.1/9-90
0x10584	DMA 1 status register (DMASR1)	R/W	0x0000_0000	9.13.1.6.2/9-92
0x10588	DMA 1 current descriptor address register (DMACDAR1)	R/W	0x0000_0000	9.13.1.6.3/9-93
0x10590	DMA 1 source address register (DMASAR1)	R/W	0x0000_0000	9.13.1.6.4/9-94
0x10598	DMA 1 destination address register (DMADAR1)	R/W	0x0000_0000	9.13.1.6.5/9-95
0x105A0	DMA 1 byte count register (DMABCR1)	R/W	0x0000_0000	9.13.1.6.6/9-95
0x105A4	DMA 1 next descriptor address register (DMANDAR1)	R/W	0x0000_0000	9.13.1.6.7/9-96
0x10600	DMA 2 mode register (DMAMR2)	R/W	0x0000_0000	9.13.1.6.1/9-90
0x10604	DMA 2 status register (DMASR2)	R/W	0x0000_0000	9.13.1.6.2/9-92

Table 9-3. Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x10608	DMA 2 current descriptor address register (DMACDAR2)	R/W	0x0000_0000	9.13.1.6.3/9-93
0x10610	DMA 2 source address register (DMASAR2)	R/W	0x0000_0000	9.13.1.6.4/9-94
0x10618	DMA 2 destination address register (DAR2)	R/W	0x0000_0000	9.13.1.6.5/9-95
0x10620	DMA 2 byte count register (DMABCR2)	R/W	0x0000_0000	9.13.1.6.6/9-95
0x10624	DMA 2 next descriptor address register (DMANDAR2)	R/W	0x0000_0000	9.13.1.6.7/9-96
0x10680	DMA 3 mode register (DMAMR3)	R/W	0x0000_0000	9.13.1.6.1/9-90
0x10684	DMA 3 status register (DMASR3)	R/W	0x0000_0000	9.13.1.6.2/9-92
0x10688	DMA 3 current descriptor address register (DMACDAR3)	R/W	0x0000_0000	9.13.1.6.3/9-93
0x10690	DMA 3 source address register (DMASAR3)	R/W	0x0000_0000	9.13.1.6.4/9-94
0x10698	DMA 3 destination address register (DMADAR3)	R/W	0x0000_0000	9.13.1.6.5/9-95
0x106A0	DMA 3 byte count register (DMABCR3)	R/W	0x0000_0000	9.13.1.6.6/9-95
0x106A4	DMA 3 next descriptor address register (DMANDAR3)	R/W	0x0000_0000	9.13.1.6.7/9-96
0x10800	PCI outbound translation address register 0 (POTAR0)	R/W	0x0000_0000	9.11.1.3/9-31
0x10808	PCI outbound base address register 0 (POBAR0)	R/W	0x0000_0000	9.11.1.4/9-32
0x10810	PCI outbound comparison mask register 0 (POCMR0)	R/W	0x0000_0000	9.11.1.5/9-33
0x10818	PCI outbound translation address register 1 (POTAR1)	R/W	0x0000_0000	9.11.1.3/9-31
0x10820	PCI outbound base address register 1 (POBAR1)	R/W	0x0000_0000	9.11.1.4/9-32
0x10828	PCI outbound comparison mask register 1 (POCMR1)	R/W	0x0000_0000	9.11.1.5/9-33
0x10830	PCI outbound translation address register 2 (POTAR2)	R/W	0x0000_0000	9.11.1.3/9-31
0x10838	PCI outbound base address register 2 (POBAR2)	R/W	0x0000_0000	9.11.1.4/9-32
0x10840	PCI outbound comparison mask register 2 (POCMR2)	R/W	0x0000_0000	9.11.1.5/9-33
0x10878	Discard timer control register (PTCR)	R/W	0x0000_0000	9.11.1.6/9-34
0x1087C	General purpose control register (GPCR)	R/W	0x0000_0000	9.11.1.7/9-34
0x10880	PCI general control register (PCI_GCR)	R/W	0x0000_0000	9.11.1.8/9-36
0x10884	Error status register (ESR)	R/W	0x0000_0000	9.11.1.9/9-36
0x10888	Error mask register (EMR)	R/W	0x0000_0FFF	9.11.1.10/9-38
0x1088C	Error control register (ECR)	R/W	0x0000_00FF	9.11.1.11/9-39
0x10890	PCI error address capture register (PCI_EACR)	R/W	0x0000_0000	9.11.1.12/9-40
0x10898	PCI error data capture register (PCI_EDCR)	R/W	0x0000_0000	9.11.1.13/9-41
0x108A0	PCI error control capture register (PCI_ECCR)	R/W	0x0000_0000	9.11.1.14/9-41
0x108D0	PCI inbound translation address register 1 (PITAR1)	R/W	0x0000_0000	9.11.1.15/9-43
0x108D8	PCI inbound base address register 1 (PIBAR1)	R/W	0x0000_0000	9.11.1.16/9-43

Table 9-3. Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x108E0	PCI inbound comparison mask register 1 (PICMR1)	R/W	0x0000_0000	9.11.1.17/9-44
0x108E8	PCI inbound translation address register 0 (PITAR0)	R/W	0x0000_0000	9.11.1.15/9-43
0x108F0	PCI inbound base address register 0 (PIBAR0)	R/W	0x0000_0000	9.11.1.16/9-43
0x108F8	PCI inbound comparison mask register 0 (PICMR0)	R/W	0x0000_0000	9.11.1.17/9-44
0x10900	PCI CFG_ADDR	R/W	Undefined	9.9.1.4.4/9-15
0x10904	PCI CFG_DATA	R/W	0x0000_0000	9.9.1.4.4/9-15
0x10908	PCI INT_ACK	R/W	Undefined	9.9.1.4.7/9-18

9.11.1.1 Message Unit (I_2O) Registers

Message unit registers are described in Section 9.12, “Message Unit (I2O).”

9.11.1.2 DMA Controller Registers

DMA registers are described in Section 9.13, “DMA Controller.”

9.11.1.3 PCI Outbound Translation Address Registers (POTAR*x*)

The PCI outbound translation address registers (POTAR*x*), shown in Figure 9-17, select the starting addresses in PCI address space for locally generated transactions that hit within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address. Refer to Section 9.10.2.2, “PCI Outbound Translation.”

	31–20	19–16
Field	—	TA
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10802 (POTAR0); 0x1081A (POTAR1); 0x10832 (POTAR2)	

	15–0
Field	TA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10800 (POTAR0); 0x10818 (POTAR1); 0x10830 (POTAR2)

Figure 9-17. PCI Outbound Translation Address Registers (POTAR*x)*

Table 9-4 describes POTAR*x*.

Table 9-4. POTAR*x* Field Descriptions

Bits	Name	Description
31–20	—	Reserved, should be cleared
19–0	Translation address	PCI address that indicates the starting point of the outbound translated address. The translation address must be aligned based on the window's size. This corresponds to bits 31–12 of a 32-bit address

9.11.1.4 PCI Outbound Base Address Registers (POBAR*x*)

The PCI outbound base address registers (POBAR*x*), shown in Figure 9-18, select the base address for the windows that are translated to the PCI address space for transactions generated by the 60x bus master or other local devices such as the DMA controller.

	31 — 20	19 — 16
Field	—	BA
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x1080A (POBAR0); 0x10822 (POBAR1); 0x1083A (POBAR2)	

	15 — 0
Field	BA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10808 (POBAR0); 0x10820 (POBAR1); 0x10838 (POBAR2)

Figure 9-18. PCI Outbound Base Address Registers (POBAR*x*)

Table 9-5 describes POBAR*x*.

Table 9-5. POBAR*x* Field Descriptions

Bits	Name	Description
31–20	—	Reserved, should be cleared
19–0	Base address	Local address that is the starting point for the outbound translation window. This corresponds to bits 31–12 of a 32-bit address

Addresses for outbound transactions are compared to the POBARs and the IMMR register. If the transaction does not fall within one of these two spaces, it is forwarded to the PCI bus without modification (see Figure 9-11). DMA-generated transactions to addresses that 'miss' the POBARs are issued (without translation) to the 60x bus (see Figure 9-13).

9.11.1.5 PCI Outbound Comparison Mask Registers (POCMR*x*)

The PCI outbound comparison mask registers (POCMR*x*), shown in Figure 9-19, define the window size to translate.

	31	30	29	28 … 20	19 … 16
Field	EN	I/O	PRE	—	CM
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x10812 (POCMR0); 0x2082A (POCMR1); 0x10842 (POCMR2)				

	15 … 0
Field	CM
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10810 (POCMR0); 0x20828 (POCMR1); 0x10840 (POCMR2)

Figure 9-19. PCI Outbound Comparison Mask Registers (POCMR*x*)

Table 9-6 describes POCMR*x*.

Table 9-6. POCMR*x* Field Descriptions

Bits	Name	Description
31	Enable	This bit enables this address translation.
30	I/O	This bit indicates that the translation is to PCI memory or PCI I/O space. 0 PCI memory 1 PCI I/O
29	Prefetchable	This bit indicates that the address space is prefetchable, so streaming can occur. 0 Not prefetchable 1 Prefetchable
28–20	—	Reserved, should be cleared
19–0	Comparison mask	Comparison mask indicates the size of the space to be translated. The value in the register represents which of the most-significant address bits to compare for a window match. Non-contiguous comparison masks will exhibit unpredictable behavior. Examples: POCMR = 0b0xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx Translation is disabled. All addresses received pass through unaltered. POCMR = 0b1xxx_xxxx_xxxx_1111_1111_1111_1111_1111 20 bits (physical address bits 31-12) are comparison masked for a 4-Kbyte window size. This is the smallest window size allowed. POCMR = 0b1xxx_xxxx_xxxx_1111_1111_1111_0000_0000 12 bits (physical address bits 31–20) for a 1-Mbyte window size.

9.11.1.6 Discard Timer Control Register (PTCR)

The discard timer control register (PTCR), shown in Figure 9-20, configures the discard timer used to put a time limit on delayed read transactions from non-prefetchable memory.

	31	30	24	23	16
Field	EN	—		PTV	
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x1087A				

	15	0
Field	PTV	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10878	

Figure 9-20. Discard Timer Control Register (PTCR)

Table 9-7 describes PTCR fields.

Table 9-7. PTCR Field Descriptions

Bits	Name	Description
31	Enable	Discard timer enable 0 Disable the discard timer. 1 Enable the discard timer.
30–24	—	Reserved
23–0	Preload timer value	Preload value for 24-bit discard timer. Delayed PCI read transactions to a non-prefetchable address space remain valid within the PCI bridge a minimum of (2^{24} – Preload Timer Value) internal clock cycles. The discard timer is used to discard delayed reads from non-prefetchable address space if the master has not repeated the transaction in *n* internal clock cycles, where *n* = (2^{24} – Preload Timer Value). Valid Preload Timer Values are in the range 0x000000–0xFFFFFE. Example: To discard a delayed completion if the PCI master has not repeated the transaction in 2^{15} PCI clocks and the internal frequency is 2 to 1 to the PCI bus. The Preload Timer Value should equal $2^{24} – 2^{16}$ (0xFF0000).

9.11.1.7 General Purpose Control Register (GPCR)

The general purpose control register (GPCR), shown in Figure 9-21, contains control bits for rerouting interrupts and adjusting the DMA controller’s 60x bandwidth.

	31	20	19	18	17	16
Field	—		DMABC		—	
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x1087E					

	15	14	13	12	1	0
Field	—	INTPCI	MCP2PCI	—		LE_MODE
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x1087C					

Figure 9-21. General Purpose Control Register (GPCR)

Table 9-8 describes GPCR fields.

Table 9-8. GPCR Field Descriptions

Bits	Name	Description
31–20	—	Reserved, should be cleared
19–18	DMABC	DMA 60x bandwidth control 00 DMA uses low 60x bandwidth. 01 DMA uses high 60x bandwidth. 10 DMA uses maximum 60x bus bandwidth. 11 DMA uses minimum 60x bandwidth. Allows breaks to be inserted in the DMA controller operation. This control may be needed to avoid starvation of other 60x masters because the PCI bridge can have higher priorities than other masters. The breaks are inserted only if some other 60x bus master requests the bus. The user should find the optimum setting by testing, arriving at the best for each specific implementation. For most systems the default value (low 60x bandwidth for the dma) will be good. Note that if the dma is the only master that needs the bus during the period of the transfer, the bandwidth is not affected.
17–15	—	Reserved, should be cleared
14	INT2PCI	Interrupt reroute to PCI 0 Interrupts are not rerouted to the PCI. Sent to the core if it is enabled or output on $\overline{\text{IRQ7}}$ if the core is disabled. 1 All SIU pending interrupts are rerouted to PCI's $\overline{\text{INTA}}$. Useful in agent mode.
13	MCP2PCI	Machine check reroute to PCI. 0 Machine check interrupts are not rerouted to the PCI. Sent to the core if it is enabled or output on $\overline{\text{IRQ0}}$ if the core is disabled 1 All machine check interrupts are rerouted to PICE’s $\overline{\text{INTA}}$. Useful in agent mode.

Table 9-8. GPCR Field Descriptions (continued)

Bits	Name	Description
12–1	—	Reserved, should be cleared
0	LE_MODE	Little-endian mode. Controls the translation of 60x-PCI and PCI-60x. Refer to Section 9.11.2.27.1, “Additional Information on Endianness,” for more details. 0 Big-endian mode. 1 Little-endian mode.

9.11.1.8 PCI General Control Register (PCI_GCR)

The PCI general control register (PCI_GCR), shown in Figure 9-22, contains a bit for controlling the PCI reset signal when in host mode.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10882

	15–1	0
Field	—	SPRST
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10880	

Figure 9-22. PCI General Control Register (PCI_GCR)

Table 9-9 describes PCI_GCR fields.

Table 9-9. PCI_GCR Field Descriptions

Bits	Name	Description
31–1	—	Reserved, should be cleared
0	Soft PCI reset	Only valid when in host mode. Allows $\overline{\text{PCI\_RST}}$ to be controlled software. Setting this bit drives the PCI reset signal high; clearing it drives the signal low.

9.11.1.9 Error Status Register (ESR)

The error status register (ESR), shown in Figure 9-23, contains status bits for various types of error conditions captured by the PCI bridge. Each status bit is set when the corresponding error condition is captured. Each bit is cleared by writing a one.

	31 16
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10886

	15 13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	I2O_ DBMC	NMI	IRA	I2O_ IPQO	I2O_ OFQO	PERR_ WR	PERR_ RD	PCI_ SERR	TAR_ ABT	NO_ RSP	DATA_ PAR_ RD	DATA_ PAR_ WR	ADDR_ PAR
Reset	0000_0000_0000_0000													
R/W	R/W													
Addr	0x10884													

Figure 9-23. Error Status Register (ESR)

Table 9-10 describes ESR fields.

Table 9-10. ESR Field Descriptions

Bits	Name	Description
31–13	—	Reserved, should be cleared
	I2O_DBMC	I_2O doorbell machine check. When a PCI-mastered write sets IDBR[31], a machine check is sent to the local processor and the event is reported in ESR[I2O_DBMC]. This bit is also set in the following cases: • An overflow condition in the inbound posted I_2O queue • An overflow condition in the outbound free I_2O queue These two interrupts can be masked in the I_2O unit.
11	NMI	General error/interrupt indication. In host mode, this bit is set when a 60x bus write transaction initiated by the PCI bridge is terminated by the assertion of $\overline{\text{TEA}}$. In agent mode, this bit is set when the GPCR[MCP2PCI] bit is set and an internal machine check interrupt (MCP) is issued by one of the MPC8272's MCP sources. Machine check and interrupt assertion is determined by ECR[11]. The reset value of ECR[11], logic zero, indicates that an interrupt will be asserted if ESR[NMI] is set (and enabled per EMR[11]).
10	IRA	Illegal register access with incorrect size
9	I2O_IPQO	I2O inbound post queue overflow
8	I2O_OFQO	I2O outbound free queue overflow
7	PCI_PERR_WR	PCI parity error received on a write
6	PCI_PERR_RD	PCI parity error received on a read
5	PCI_SERR	PCI $\overline{\text{SERR}}$ received
4	PCI_TAR_ABT	PCI target abort
3	PCI_NO_RSP	PCI no response (no $\overline{\text{DEVSEL}}$; master abort)

Table 9-10. ESR Field Descriptions (continued)

Bits	Name	Description
2	PCI_DATA_PAR_RD	PCI read data parity error
1	PCI_DATA_PAR_WR	PCI write data parity error
0	PCI_ADDR_PAR	PCI address parity error (read or write)

9.11.1.10 Error Mask Register (EMR)

The error mask register (EMR) register, shown in Figure 9-24, enables the IOU to assert an interrupt or a machine check for the various types of error conditions listed in Table 9-10. Each mask bit is active high. That is, if a bit value is zero, an interrupt or machine check is not asserted for the corresponding error condition.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x1088A

	15–13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—	I2O_ DBMC	NMI	IRA	I2O_ IPQO	I2O_ OFQO	PERR_ WR	PERR_ RD	PCI_ SERR	TAR_ ABT	NO_ RSP	DATA_ PAR_ RD	DATA_ PAR_ WR	ADDR_ PAR
Reset	0000_1111_1111_1111													
R/W	R/W													
Addr	0x10888													

Figure 9-24. Error Mask Register (EMR)

Table 9-11 describes EMR fields.

Table 9-11. EMR Field Descriptions

Bits	Name	Description
31–13	—	Reserved, should be cleared
12	I2O_DBMC	I_2O doorbell machine check 0 Machine check is not enabled. 1 Machine check is enabled.
11	NMI	General error/interrupt indication
10	IRA	Illegal register access with incorrect size
9	I2O_IPQO	I2O inbound post queue overflow
8	I2O_OFQO	I2O outbound free queue overflow

Table 9-11. EMR Field Descriptions (continued)

Bits	Name	Description
7	PCI_PERR_WR	PCI parity error received on a write. The MPC8272 sinks PERR. This error is only a function of data.
6	PCI_PERR_RD	PCI parity error received on a read. The MPC8272 sinks PERR. This error is only a function of data.
5	PCI_SERR	PCI $\overline{\text{SERR}}$ received
4	PCI_TAR_ABT	PCI target abort
3	PCI_NO_RSP	PCI no response (no $\overline{\text{DEVSEL}}$; master abort)
2	PCI_DATA_PAR_RD	PCI read data parity error. The MPC8272 sources PERR. This error is only a function of data.
1	PCI_DATA_PAR_WR	PCI write data parity error. The MPC8272 sources PERR. This error is only a function of data.
0	PCI_ADDR_PAR	PCI address parity error (read or write)

9.11.1.11 Error Control Register (ECR)

The error control register (ECR) register, shown in Figure 9-25, determines whether the IOU asserts an interrupt or a machine check for the error conditions listed in Table 9-10. The IOU asserts an interrupt or machine check only if the mask bit for the error condition (refer to Table 9-11) is set. Each bit is defined as follows:

- Zero—The IOU issues an interrupt upon the error condition.
- One—The IOU issues a machine check upon the error condition.

	31	16
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x1088E	

	15	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	—		I2O_ DBMC	NMI	IRA	I2O_ IPQO	I2O_ OFQO	PERR_ WR	PERR_ RD	SERR	TAR_ ABT	NO_ RSP	DATA_ PAR_ RD	DATA_ PAR_ WR	ADDR_ PAR
Reset	0000_0000_1111_1111														
R/W	R/W														
Addr	0x1088C														

Figure 9-25. Error Control Register (ECR)

Table 9-12 describes ECR fields.

Table 9-12. ECR Field Descriptions

Bits	Name	Description
31–13	—	Reserved, should be cleared
12	I2O_DBMC	I$_2$O doorbell machine check 0 ESR[I2O_DBMC] causes an interrupt. 1 ESR[I2O_DBMC] (if enabled) causes a machine check.
11	NMI	General error/interrupt indication
10	IRA	Illegal register access with incorrect size
9	I2O_IPQO	I2O inbound post queue overflow
8	I2O_OFQO	I2O outbound free queue overflow
7	PCI_PERR_WR	PCI parity error received on a write
6	PCI_PERR_RD	PCI parity error received on a read
5	PCI_SERR	PCI $\overline{\text{SERR}}$ received
4	PCI_TAR_ABT	PCI target abort
3	PCI_NO_RSP	PCI no response (no $\overline{\text{DEVSEL}}$; master abort)
2	PCI_DATA_PAR_RD	PCI read data parity error
1	PCI_DATA_PAR_WR	PCI write data parity error
0	PCI_ADDR_PAR	PCI address parity error (read or write)

9.11.1.12 PCI Error Address Capture Register (PCI_EACR)

The PCI error address capture register (PCI_EACR), shown in Figure 9-26, stores the address associated with the first PCI error captured.

	31 … 16
Field	PCI_EAR
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10892

	15 … 0
Field	PCI_EAR
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10890

Figure 9-26. PCI Error Address Capture Register (PCI_EACR)

Table 9-13 describes PCI_EACR fields.

Table 9-13. PCI_EACR Field Descriptions

Bits	Name	Description
31–0	PCI_EAR	The address associated with the first error captured.

9.11.1.13 PCI Error Data Capture Register (PCI_EDCR)

The PCI error data capture register (PCI_EDCR), shown in Figure 9-27, stores the data associated with the first PCI error captured.

	31	16
Field	PCI_EDR	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x1089A	

	15	0
Field	PCI_EDR	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10898	

Figure 9-27. PCI Error Data Capture Register (PCI_EDCR)

Table 9-14 describes PCI_EDCR fields.

Table 9-14. PCI_EDCR Field Description

Bits	Name	Description
31–0	PCI_EDR	The data associated with the first error captured

9.11.1.14 PCI Error Control Capture Register (PCI_ECCR)

The PCI error control capture register (PCI_ECCR), shown in Figure 9-28, stores information associated with the first PCI error captured.

	31	30 28	27 24	23 22	21 20	19 16
Field	—	FET	BN	—	TS	ES
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x108A2					

	15 12	11 8	7 4	3 2	1	0
Field	CMD	—	BE	—	PB	VI
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x108A0					

Figure 9-28. PCI Error Control Capture Register (PCI_ECCR)

Table 9-15 describes PCI_ECCR fields.

Table 9-15. PCI_ECCR Field Descriptions

Bits	Name	Description
31	—	Reserved, should be cleared
30–28	First error type	Type of first PCI error captured. This field is the bit index of the error type in Table 9-10. For example, a value of 0b101 indicates a PCI $\overline{\text{SERR}}$ received condition while a value of 0b010 indicates a PCI read data parity error.
27–24	Beat number	32-bit data beat number for data parity error (data parity error only) 0000 1 0001 2 … 0111 8 1000 overflow (transaction larger than one cache line)
23–22	—	Reserved, should be cleared
21–20	Transaction size	This is the size of the transaction in double words (4 bytes) (the PCI bridge as master only) 00 4 double words 01 1 double word 10 2 double words 11 3 double words
19–16	Error source	The source of the PCI transaction 0000 External master 0101 DMA All others are reserved.
15–12	Command	PCI command
11–8	—	Reserved, should be cleared
7–4	Byte enables	PCI byte enables.
3–2	—	Reserved, should be cleared

Table 9-15. PCI_ECCR Field Descriptions (continued)

Bits	Name	Description
1	Parity bit	Parity bit for PCI bus data word
0	Valid info	When this bit is set, the PCI bus error capture registers (PCI_EACR, PCI_EDCR, and PCI_ECCR) contain valid information. Writing ‘0’ to this bit enables the capture of a new error in the PCI bus error capture registers (PCI_EACR, PCI_EDCR, and PCI_ECCR).

9.11.1.15 PCI Inbound Translation Address Registers (PITAR*x*)

The PCI inbound translation address registers (PITAR*x*), shown in Figure 9-29, select the base addresses in the 60x address space of the translation windows for transactions generated by the master on the PCI bus. The new translated address is created by concatenating the transaction offset to this base address. Refer to Section 9.10.2.1, “PCI Inbound Translation.”

	31 20	19 16
Field	—	TA
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x108EC (PITAR0); 0x108D2 (PITAR1)	

	15 0
Field	TA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x108EA (PITAR0); 0x108D0 (PITAR1)

Figure 9-29. PCI Inbound Translation Address Registers (PITAR*x*)

Table 9-16 describes PITAR*x*.

Table 9-16. PITAR*x* Field Descriptions

Bits	Name	Description
31–20	—	Reserved, should be cleared
19–0	Translation address	60x address that indicates the starting point of the inbound translated address. The translation address must be aligned based on the window’s size. This corresponds to bits 31–12 of a 32-bit address

9.11.1.16 PCI Inbound Base Address Registers (PIBAR*x*)

The PCI inbound base address registers (PIBAR*x*), shown in Figure 9-30, select the starting addresses (in PCI memory space) of the windows to be translated. These registers are tied to the GPLABAR*x* registers; see Section 9.11.2.14, “General-Purpose Local Access Base Address Registers (GPLABARx).” A change in a PIBAR*x* register causes a change in the GPLABAR*x* in the base address bits that are non-masked by PICMR*x*, and vice versa.

The system host is responsible for the configuration of the base address by writing to GPLABAR*x*; therefore, in PCI agent mode, the PIBAR*x* registers should be read-only. However, if the PCI bridge is defined as the PCI host, it may be easier to configure its own inbound base address by writing directly to the PIBAR*x* registers.

	31 … 20	19 … 16
Field	—	BA
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x108F2 (PITAR0); 0x108DA (PITAR0)	

	15 … 0
Field	BA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x108F0 (PITAR0); 0x108D8 (PITAR0)

Figure 9-30. PCI Inbound Base Address Registers (PIBAR*x*)

Table 9-17 describes PIBAR*x*.

9.11.1.17 PCI Inbound Comparison Mask Registers (PICMR*x*)

Table 9-17. PIBAR*x* Field Descriptions

Bits	Name	Description
31–20	—	Reserved, should be cleared
19–0	Base address	PCI address that is the starting point for the inbound translation window. This corresponds to bits 31–12 of a 32-bit address

The PCI inbound comparison mask registers (PICMR*x*), shown in Figure 9-31, defines the inbound window's size. In PCI agent mode, this register should be initialized (either by the core or by the CP's automatic EEPROM load) before the AGENT_CFG_LOCK bit (see Section 9.11.2.22, "PCI Bus Function Register") can be cleared to enable the host to configure the device. Some of the fields of this registers are tied to the GPLABAR*x* registers; see Section 9.11.2.14, "General-Purpose Local Access Base Address Registers (GPLABARx)."

	31	30	29	28	20	19	16
Field	EN	NO_ SNOOP_ EN	PRE	—		CM	
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x108FA (PICMR0); 0x108E2 (PICMR1)						

	15	0
Field	CM	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x108F8 (PICMR0); 0x108E0 (PICMR1)	

Figure 9-31. PCI Inbound Comparison Mask Registers (PICMR*x)*

Table 9-18 describes PICMR*x*.

Table 9-18. PICMR*x* Field Descriptions

Bits	Name	Description
31	Enable	Setting this bit enables address translation.
30	NO_SNOOP_EN	Controls whether the PCI bridge generates snoop transactions on the 60x bus for PCI-to-60x memory transactions that hit in this address translation window. Disabling snooping is a performance enhancement for systems that do not need to maintain coherency on system memory accesses by PCI. 0 Snooping is enabled. 1 Snooping is disabled.
29	Prefetchable	Indicates whether the address space is prefetchable so that streaming can occur. 0 Not prefetchable 1 Prefetchable
28–20	—	Reserved, should be cleared.
19–0	Comparison mask	Indicates the size of the space to be translated. The value in the register represents which of the most-significant address bits to compare for a window match. Non-contiguous comparison mask bits cause unpredictable behavior. Examples: PICMR = 0b0xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx Inbound window is disabled. PICMR = 0b1xxx_xxxx_xxxx_1111_1111_1111_1111_1111 The mask is 20 bits (physical address bits 31–12) that corresponds to a 4-Kbyte window size. This is the smallest window size allowed. PICMR = 0b1xxx_xxxx_xxxx_1111_1111_1111_0000_0000 The mask is 12 bits (physical address bits 31-20) which corresponds to a 1-Mbyte window size.

9.11.2 PCI Bridge Configuration Registers

The *PCI Local Bus Specification* defines the configuration registers from 0x00 through 0x3F. Additionally, the PCI bridge specifies these additional registers: the PCI function register (at offset 0x44), the PCI arbiter control register (at offset 0x46), and the PCI Hot Swap register block (at offset 0x48). Table 9-19 and Figure 9-32 shows the PCI configuration registers provided by the PCI bridge for the PCI bus.

Note the following sections that apply to all PCI configuration registers (they appear immediately after the descriptions of individual registers):

- Section 9.11.2.26, “PCI Configuration Register Access from the Core,” on page 9-63
- Section 9.11.2.27, “PCI Configuration Register Access in Big-Endian Mode,” on page 9-63
- Section 9.11.2.28, “Initializing the PCI Configuration Registers,” on page 9-65

Table 9-19. PCI Bridge PCI Configuration Registers

Address (offset)	Register	Access	Reset	Section/Page
00	Vendor ID	R	0x1057	9.11.2.1/9-47
02	Device ID	R	0x18C0	9.11.2.2/9-48
04	PCI command	R/W	Mode-dependent	9.11.2.3/9-48
06	PCI status	Read/bit-reset	0x00B0	9.11.2.4/9-49
08	Revision ID	R	Rev-dependent	9.11.2.5/9-51
09	Standard programming interface	R	Mode-dependent	9.11.2.6/9-51
0A	Subclass code	R	0x00	9.11.2.7/9-52
0B	Class code	R	Mode-dependent	9.11.2.8/9-52
0C	Cache line size	R/W	0x00	9.11.2.9/9-53
0D	Latency timer	R/W	0x00	9.11.2.10/9-53
0E	Header type	R	0x00	9.11.2.11/9-54
0F	BIST control	R	0x00	9.11.2.12/9-54
10	PIMMR base address register	R/W	0x*nnnn*_0000	9.11.2.13/9-54
14	GPL base address register 0	R/W	0x0000_0000	9.11.2.14/9-55
18	GPL base address register 1	R/W	0x0000_0000	9.11.2.14/9-55
1C	Reserved	—	—	—
2C	Sub system vendor ID	R/W	0x0000	9.11.2.15/9-56
2E	Sub system device ID	R/W	0x0000	9.11.2.16/9-57
30	Reserved	—	—	—
34	Capabilities pointer	R	0x48	9.11.2.17/9-57
35	Reserved	—	—	—
3C	Interrupt line	R/W	0x00	9.11.2.18/9-58
3D	Interrupt pin	R	0x01	9.11.2.19/9-58

Table 9-19. PCI Bridge PCI Configuration Registers (continued)

Address (offset)	Register	Access	Reset	Section/Page
3E	MIN GNT	R	0x00	9.11.2.20/9-59
3F	MAX LAT	R	0x00	9.11.2.21/9-59
40	Reserved	—	—	—
44	PCI function	R/W	0x0000	9.11.2.22/9-60
46	PCI arbiter control register	R/W	Mode-dependent	9.11.2.23/9-61
48	Hot swap register block	R/W	0x00nn_0006	9.11.2.24/9-61 9.11.2.25/9-62

<table>
<tr><th>Address offset (Hex)</th><th colspan="8"></th></tr>
<tr><td>00</td><td colspan="4">Device ID (0x18C0)</td><td colspan="4">Vendor ID (0x1057)</td></tr>
<tr><td>04</td><td colspan="4">PCI Status</td><td colspan="4">PCI Command</td></tr>
<tr><td>08</td><td colspan="2">Class Code</td><td colspan="2">Subclass Code</td><td colspan="2">Standard Programming</td><td colspan="2">Revision ID</td></tr>
<tr><td>0C</td><td colspan="2">BIST Control</td><td colspan="2">Header Type</td><td colspan="2">Latency Timer</td><td colspan="2">Cache Line Size</td></tr>
<tr><td>10</td><td colspan="8">PIMMR Base Address Register</td></tr>
<tr><td>14</td><td colspan="8">GPLA Base Address Register 0</td></tr>
<tr><td>18</td><td colspan="8">GPLA Base Address Register 1</td></tr>
<tr><td></td><td></td><td colspan="2">:</td><td colspan="2">—</td><td colspan="2">:</td><td></td></tr>
<tr><td>2C</td><td colspan="4">Subsystem ID</td><td colspan="4">Subsystem Vendor ID</td></tr>
<tr><td></td><td></td><td colspan="2">:</td><td colspan="2">—</td><td colspan="2">:</td><td></td></tr>
<tr><td>34</td><td colspan="6">—</td><td colspan="2">Capabilities Pointer</td></tr>
<tr><td>38</td><td colspan="8">—</td></tr>
<tr><td>3C</td><td colspan="2">MAX LAT</td><td colspan="2">MIN GNT</td><td colspan="2">Interrupt Pin</td><td colspan="2">Interrupt Line</td></tr>
<tr><td>40</td><td colspan="8">—</td></tr>
<tr><td>44</td><td colspan="4">PCI Arbiter Control</td><td colspan="4">PCI Function</td></tr>
<tr><td>48</td><td colspan="2"></td><td colspan="2">Hot Swap CSR</td><td colspan="2"></td><td colspan="2">Hot Swap Capability ID</td></tr>
</table>

Figure 9-32. PCI Bridge PCI Configuration Registers

The PCI configuration registers are accessible from the core through an indirect method discussed in Section 9.11.2.26, “PCI Configuration Register Access from the Core.” The registers are accessible from the PCI bus through the PCI configuration transaction when the PCI bridge is in agent mode.

The following sections describe the individual PCI configuration registers.

9.11.2.1 Vendor ID Register

Figure 9-33 and Table 9-20 describe the vendor ID register.

	15 … 0
Field	VID
Reset	0001_0000_0101_0111
R/W	R
Addr	0x00

Figure 9-33. Vendor ID Register

Table 9-20. Vendor ID Register Description

Bits	Name	Description
15–0	Vendor ID	Identifies the manufacturer of the device (0x1057 = Freescale)

9.11.2.2 Device ID Register

Figure 9-34 and Table 9-21 describes the device ID register.

	15 … 0
Field	DID
Reset	0001_1000_1100_0000
R/W	R
Addr	0x02

Figure 9-34. Device ID Register

Table 9-21. Device ID Register Description

Bits	Name	Description
15–0	Device ID	Identifies the particular device (0x18C1 = MPC8272)

9.11.2.3 PCI Bus Command Register

Figure 9-35 and Table 9-22 describe the PCI bus command register that provides control over the ability to generate and respond to PCI cycles.

	15 … 10	9	8	7	6	5	4	3	2	1	0
Field	—	FB-B	SERR	—	PERRR	—	MWI	SC	BM	MEM	I/O
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x04										

Figure 9-35. PCI Bus Command Register

Table 9-22. PCI Bus Command Register Description

Bits	Name	Description
15–10	—	Reserved, should be cleared
9	Fast back-to-back	Hardwired to 0, indicating that the PCI bridge as a master does not run fast back-to-back transactions
8	SERR	Controls the $\overline{\text{SERR}}$ driver of the PCI bridge. This bit (and bit 6) must be set to report address parity errors. 0 Disables the $\overline{\text{SERR}}$ driver 1 Enables the $\overline{\text{SERR}}$ driver
7	—	Reserved, should be cleared
6	Parity error response	Controls whether the PCI bridge responds to parity errors on the PCI bus 0 Parity errors are ignored and normal operation continues. 1 Action is taken on a parity error.
5	—	Reserved, should be cleared
4	Memory-write-and-invalidate	Hardwired to 0, indicating that the PCI bridge acting as a master does not generate the memory-write-and-invalidate command. The PCI bridge generates a memory-write command instead.
3	Special cycles	Hardwired to 0, indicating that the PCI bridge as a target ignores all special-cycle commands
2	Bus master	Controls whether the PCI bridge can act as a master on the PCI bus. This bit is cleared if the PCI bridge is powered-up as an agent device and is set if it is powered-up as a host bridge device. 0 Disables the ability to generate PCI accesses. In host bridge mode, read transactions return undefined data and write transactions lose data. In agent mode, transactions are held until this bit is enabled. 1 Enables the PCI bridge to behave as a PCI bus master
1	Memory space	Controls whether the PCI bridge as a target responds to memory accesses. 0 The PCI bridge does not respond to PCI memory space accesses. 1 The PCI bridge responds to PCI memory space accesses.
0	I/O space	Hardwired to 0, indicating that the PCI bridge as a target does not respond to PCI I/O space accesses.

9.11.2.4 PCI Bus Status Register

The PCI bus status register, shown in Figure 9-36, is used to record status information for PCI bus-related events. Only 2-byte accesses to address offset 0x06 are allowed.

Reads to this register behave normally. Writes are slightly different in that bits can be cleared, but not set. A bit is cleared whenever the register is written, and the data in the corresponding bit location is set. For example, to clear bit 14 and not affect any other bits in the register, write the value 0b0100_0000_0000_0000 to the register.

	15	14	13	12	11	10 9	8	7	6	5	4	3 0
Field	DPERR	SSERR	RM-A	RT-A	ST-A	$\overline{\text{DEVSEL}}$_T	DPD	FB-BC	—	66MHzC	CL	—
Reset	0000_0000_1011_0000											
R/W	R/W									R	R/W	
Addr	0x06											

Figure 9-36. PCI Bus Status Register

Table 9-23 describes the PCI bus status register fields.

Table 9-23. PCI Bus Status Register Description

Bits	Name	Description
15	Detected parity error	Set whenever the PCI bridge detects a parity error on the PCI bus, even if parity error handling is disabled (as controlled by bit 6 in the PCI bus command register)
14	Signaled system error	Set whenever the PCI bridge asserts $\overline{\text{SERR}}$ on the PCI bus
13	Received master-abort	Set whenever the PCI bridge, acting as the PCI master on the PCI bus, terminates a transaction (except for a special-cycle) using master-abort
12	Received target-abort	Set whenever a PCI bridge-initiated transaction on the PCI bus is terminated by a target-abort
11	Signaled target-abort	Set whenever the PCI bridge, acting as the PCI target on the PCI bus, issues a target-abort to a PCI master
10–9	$\overline{\text{DEVSEL}}$ timing	Hardwired to 0b00, indicating that the PCI bridge uses fast device-select timing on the PCI bus
8	Data parity detected	Set upon detecting a data parity error on the PCI bus. Three conditions must be met for this bit to be set: • The PCI bridge detects a parity error. • The PCI bridge is acting as the bus master for the operation in which the error occurred. • Bit 6 in the PCI bus command register is set.
7	Fast back-to-back capable	Hardwired to 1, indicating that the PCI bridge as a target is capable of accepting fast back-to-back transactions
6	—	Reserved, should be cleared
5	66-MHz capable	This bit is read-only and indicates that the PCI bridge is capable of 66-MHz PCI bus operation on the PCI bus.
4	Capabilities list	Hardwired to 1, indicating that the PCI bridge implements new capabilities on the PCI bus
3–0	—	Reserved, should be cleared

9.11.2.5 Revision ID Register

Figure 9-37 and Table 9-24 describe the revision ID register.

	7 … 0
Field	RID
Reset	Refer to Table 9-24.
R/W	R
Addr	0x08

Figure 9-37. Revision ID Register

Table 9-24. Revision ID Register Description

Bits	Name	Reset Value	Description
7–0	Revision ID	Revision Dependent	Specifies a device-specific revision code for the MPC8272 (assigned by Freescale). Revision ID = 0x11 for .25-micron revisions A.0, B.1, and C.0. Revision ID = 0x10 for .13-micron PQ27e devices.

9.11.2.6 PCI Bus Programming Interface Register

Figure 9-38 and Table 9-25 describe the PCI bus programming interface register.

	7 … 0
Field	PI
Reset	Refer to Table 9-25.
R/W	R
Addr	0x09

Figure 9-38. PCI Bus Programming Interface Register

Table 9-25. PCI Bus Programming Interface Register Description

Bits	Name	Description
7–0	Programming interface	0x00 When the PCI bridge is configured as host bridge 0x01 When the PCI bridge is configured as a peripheral device to indicate the programming model supports the I_2O interface

9.11.2.7 Subclass Code Register

Figure 9-39 and Table 9-26 describe the subclass code register.

	7 — 0
Field	SC
Reset	0000_0000
R/W	R
Addr	0x0A

Figure 9-39. Subclass Code Register

Table 9-26. Subclass Code Register Description

Bits	Name	Description
7–0	Subclass code	Identifies more specifically the function of the PCI bridge (0x00 = host bridge and 0x80 = agent).

9.11.2.8 PCI Bus Base Class Code Register

Figure 9-40 and Table 9-27 describe the PCI bus class code register.

	7 — 0
Field	BCC
Reset	Refer to Table 9-27.
R/W	R
Addr	0x0B

Figure 9-40. PCI Bus Base Class Code Register

Table 9-27. PCI Bus Base Class Code Register Description

Bits	Name	Description
7–0	Base class code	0x06 When the PCI bridge is configured as a host bridge to indicate “Host Bridge” 0x0E When the PCI bridge is configured as a target device to indicate the device is an agent and is I_2O capable

NOTE: I_2O Compliancy

When configured as a PCI agent device, the value of the interface, subclass code, and base class code registers are 0x01, 0x00, and 0x0E respectively, indicating that the MPC8272 supports the I_2O protocol. The user should note that the I_2O support is not fully standard compliant.

9.11.2.9 PCI Bus Cache Line Size Register

Figure 9-41 and Table 9-28 describe the PCI bus cache line size register.

	7	0
Field	CLS	
Reset	0000_0000	
R/W	R/W	
Addr	0x0C	

Figure 9-41. PCI Bus Cache Line Size Register

Table 9-28. PCI Bus Cache Line Size Register Description

Bits	Name	Description
7–0	Cache line size	Represents the cache line size of the system in terms of 32-bit words (eight 32-bit words = 32 bytes). This register is read-write; however, an attempt to program this register to any value other than eight results in it being cleared.

9.11.2.10 PCI Bus Latency Timer Register

Figure 9-42 and Table 9-29 describe the PCI bus latency timer register.

	7	3	2	0
Field	LT		LT	
Reset	0000_0000			
R/W	R/W		R	
Addr	0x0D			

Figure 9-42. PCI Bus Latency Timer Register

Table 9-29. PCI Bus Latency Timer Register Description

Bits	Name	Description
7–3	Latency timer	Represents the maximum number of PCI clocks that the device, when mastering a transaction, holds the bus after PCI bus grant has been negated. The value is in PCI clocks. Refer to the PCI 2.2 specification for the rules by which the PCI bus interface unit completes transactions when the timer has expired.
2–0		Read-only least-significant bits of the latency timer. (The latency timer value is programmed in multiples of eight.)

9.11.2.11 Header Type Register

Figure 9-43 and Table 9-30 describe the header type register.

	7	6 0
Field	MD	HT
Reset	0000_0000	
R/W	R	
Addr	0x0E	

Figure 9-43. Header Type Register

Table 9-30. Header Type Register Description

Bits	Name	Description
7	Multifunction device	The PCI bridge is not a multifunction PCI device.
6–0	Header type	Identifies the layout of bytes 0x10–0x3F of the configuration address space

9.11.2.12 BIST Control Register

Figure 9-44 and Table 9-31 describe the BIST control register.

	7 0
Field	BIST_CTRL
Reset	0000_0000
R/W	R
Addr	0x0F

Figure 9-44. BIST Control Register

Table 9-31. BIST Control Register Description

Bits	Name	Description
7–0	BIST control	Optional register for control and status of built-in self test (BIST)

9.11.2.13 PCI Bus Internal Memory-Mapped Registers Base Address Register (PIMMRBAR)

In agent mode, the PCI bridge provides one base address register called the PCI bus internal memory-mapped registers base address register (PIMMRBAR) to allow a host processor access to the MPC8272’s internal memory-mapped registers. Transactions from PCI that 'hit' the PIMMRBAR are translated to the IMMR and sent to the logic that controls the internal memory-mapped registers. PIMMRBAR is shown in Figure 9-45.

	31 — 17	16
Field	BA	BA
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x12	

	15 — 4	3	2 — 1	0
Field	BA	PRE	T	MSI
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x10			

Figure 9-45. PCI Bus Internal Memory-Mapped Registers Base Address Register (PIMMRBAR)

Table 9-32 describes PIMMRBAR fields.

Table 9-32. PIMMRBAR Field Descriptions

Bits	Name	Description
31–17	Base address	Indicates the base address for the inbound configuration window
16–4	Base address	Hardwired to zeros, indicating that the PCI bridge requires a 128-Kbyte space for the configuration registers
3	Prefetchable	Hardwired to 0 to indicate that this address region is not prefetchable
2–1	Type	Hardwired to 00 to indicate that the address can be located anywhere in 32-bit address space
0	Memory space indicator	Address is mapped to memory space

9.11.2.14 General-Purpose Local Access Base Address Registers (GPLABAR*x*)

Two general-purpose local access base address registers (GPLABAR*x*) are provided to allow access to local memory space. These registers are closely tied to PIBAR*x* and PICMR*x* (see Section 9.11.1.16, “PCI Inbound Base Address Registers (PIBARx),” and Section 9.11.1.17, “PCI Inbound Comparison Mask Registers (PICMRx)”). A write to GPLABAR*x* causes a write to PIBAR*x* but only to the bits allowed by the PICMR*x* mask. Similarly, a write to PIBAR*x* causes a write to GPLABAR*x* of the non-masked bits of the base address. GPLABAR*x* is shown in Figure 9-46.

	31 16
Field	BA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x16 (GPLABAR0); 0x1A (GPLABAR1)

	15 12	11 4	3	2 1	0
Field	BA	BA	PRE	T	MSI
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x14 (GPLABAR0); 0x18 (GPLABAR1)				

Figure 9-46. General Purpose Local Access Base Address Registers (GPLABAR*x*)

Table 9-33 describes GPLABAR*x* fields.

Table 9-33. GPLABAR*x* Field Descriptions

Bits	Name	Description
31–12	Base address	Represents the base address for the inbound GPLA memory window. The number of upper bits that the PCI bridge allows to be writable is selected through the PICMR; see Section 9.11.1.17, “PCI Inbound Comparison Mask Registers (PICMRx).”
11–4		Hardwired to zeros. (The minimum window size allowed is 4K.)
3	Prefetchable	Corresponds to the prefetchable bit in the PICMR; see Section 9.11.1.17, “PCI Inbound Comparison Mask Registers (PICMRx).”
2–1	Type	Hardwired to 00 to indicate that the address can be located anywhere in 32-bit address space
0	Memory space indicator	Address is mapped to memory space (hardwired to 0)

9.11.2.15 Subsystem Vendor ID Register

Figure 9-47 and Table 9-34 describe the subsystem vendor ID register.

	15 0
Field	SVID
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x2C

Figure 9-47. Subsystem Vendor ID Register

Table 9-34. Subsystem Vendor ID Register Description

Bits	Name	Description
15–0	Vendor ID	Identifies the add-in board or subsystem where the PCI device resides

9.11.2.16 Subsystem Device ID Register

Figure 9-48 and Table 9-35 describe the subsystem ID register.

	15	0
Field	SDID	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x2E	

Figure 9-48. Subsystem Device ID Register

Table 9-35. Subsystem Device ID Description Register

Bits	Name	Description
15–0	Subsystem ID	Identifies the add-in board or subsystem where the PCI device resides

9.11.2.17 PCI Bus Capabilities Pointer Register

Figure 9-49 and Table 9-36 describe the PCI bus capabilities pointer register.

	7	0
Field	CP	
Reset	0100_1000	
R/W	R	
Addr	0x34	

Figure 9-49. PCI Bus Capabilities Pointer Register

Table 9-36. PCI Bus Capabilities Pointer Register Description

Bits	Name	Description
7–0	Capabilities pointer	Specifies the byte offset in the configuration space containing the first item in the capabilities list

9.11.2.18 PCI Bus Interrupt Line Register

Figure 9-50 and Table 9-37 describes the PCI bus interrupt line register.

<table>
<tr><th></th><th>7</th><th>0</th></tr>
<tr><td>Field</td><td colspan="2">IL</td></tr>
<tr><td>Reset</td><td colspan="2">0000_0000</td></tr>
<tr><td>R/W</td><td colspan="2">R/W</td></tr>
<tr><td>Addr</td><td colspan="2">0x3C</td></tr>
</table>

Figure 9-50. PCI Bus Interrupt Line Register

Table 9-37. PCI Bus Interrupt Line Register Description

Bits	Name	Description
7–0	Interrupt line	Contains the interrupt routing information. Software can use this register to hold information regarding which input of the system interrupt controller the $\overline{\text{INTA}}$ signal is attached to. Values in this register are specific to the system architecture.

9.11.2.19 PCI Bus Interrupt Pin Register

Figure 9-51 and Table 9-38 describe the PCI bus interrupt pin register.

<table>
<tr><th></th><th>7</th><th>0</th></tr>
<tr><td>Field</td><td colspan="2">IP</td></tr>
<tr><td>Reset</td><td colspan="2">0000_0001</td></tr>
<tr><td>R/W</td><td colspan="2">R</td></tr>
<tr><td>Addr</td><td colspan="2">0x3D</td></tr>
</table>

Figure 9-51. PCI Bus Interrupt Pin Register

Table 9-38. PCI Bus Interrupt Pin Register Description

Bits	Name	Description
7–0	Interrupt pin	Indicates which interrupt pin the device (or function) uses (0x01 = $\overline{\text{INTA}}$)

9.11.2.20 PCI Bus MIN GNT

Figure 9-52 and Table 9-39 describes the PCI bus MIN GNT register.

	7	0
Field	MIN GNT	
Reset	0000_0000	
R/W	R	
Addr	0x3E	

Figure 9-52. PCI Bus MIN GNT

Table 9-39. PCI Bus MIN GNT Description

Bits	Name	Description
7–0	MIN GNT	Specifies the length of the device’s burst period. The value 0x00 indicates that the PCI bridge has no major requirements for the settings of latency timers.

9.11.2.21 PCI Bus MAX LAT

Figure 9-53 and Table 9-40 describe the PCI bus MAX LAT register.

	7	0
Field	MAX LAT	
Reset	0000_0000	
R/W	R	
Addr	0x3F	

Figure 9-53. PCI Bus MAX LAT

Table 9-40. PCI Bus MAX LAT Description

Bits	Name	Description
7–0	MAX LAT	Specifies how often the device needs to gain access to the PCI bus. The value 0x00 indicates that the PCI bridge has no major requirements for the settings of latency timers.

9.11.2.22 PCI Bus Function Register

The PCI bus function register, shown in Figure 9-54, is used to determine the configuration of the PCI bus interface.

<table>
<tr><td></td><td>15</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Field</td><td colspan="2">—</td><td>CFG_LOCK</td><td colspan="2">—</td><td>TRGT_
LATENCY_DIS</td><td>MSTR_
LATENCY_DIS</td><td>PCI_HA</td></tr>
<tr><td>Reset</td><td colspan="8">0000_0000_0010_0000</td></tr>
<tr><td>R/W</td><td colspan="7">R/W</td><td>R</td></tr>
<tr><td>Addr</td><td colspan="8">0x44</td></tr>
</table>

Figure 9-54. PCI Bus Function Register

Table 9-41 describes PCI bus function register fields.

Table 9-41. PCI Bus Function Register Field Descriptions

Bits	Name	Description
15–6	—	Reserved, should be cleared
5	CFG_LOCK	Agent mode: Setting CFG_LOCK prevents an external PCI master from accessing the configuration space while the 60x bus is doing internal configuration. It is explicitly set and cleared by the 60x bus. 0 PCI bridge accepts accesses to the PCI configuration space or the internal memory-mapped configuration space. 1 PCI bridge retries all accesses to the PCI configuration space or the internal memory-mapped configuration space. Host mode: the PCI configuration space is not accessible from the PCI side when the device is in host mode; therefore, this bit applies only for the internal memory-mapped configuration space. 0 PCI bridge accepts accesses to the internal memory-mapped configuration space. 1 PCI bridge retries all accesses to the internal memory-mapped configuration space.
4–3	—	Reserved, should be cleared
2	TRGT_LATENCY_DIS	Target latency time-out disable. Controls whether the PCI bridge as a target time-outs when the first data phase of a transaction has not completed in 16 PCI cycles. 0 Target latency time-out enabled 1 Target latency time-out disabled
1	MSTR_LATENCY_DIS	Master latency timer disable. Controls whether the PCI bridge as a master ends a transaction after the expiration of the master latency timer. See Section 9.11.2.10, “PCI Bus Latency Timer Register.” 0 Master latency timer enabled 1 Master latency timer disabled
0	PCI_HA	Set or cleared by a power-On configuration bit on power-up and is read only. 0 PCI interface is in host mode 1 PCI interface is in agent mode

9.11.2.23 PCI Bus Arbiter Configuration Register

The PCI bus arbiter configuration register, shown in Figure 9-55, is used to determine the configuration of the PCI bus arbiter. Only 1-byte or 2-byte accesses to address offset 0x46 are allowed.

	15	14	13 7	6 4	3 1	0
Field	PCI_ ARB_DIS	PM	—	PCI_BUSMP	—	PCI_ BRIDGE MP
Reset	0[1]000_0000_0000_0000					
R/W	R/W					
Addr	0x46					

[1] Reset value determined by PIC_CFG[1] pin value after hard reset. Refer to Table 9-42.

Figure 9-55. PCI Bus Arbiter Configuration Register

Table 9-42 describes the PCI bus arbiter configuration register fields.

Table 9-42. PCI Bus Arbiter Configuration Register Field Description

Bit	Name	Description
15	PCI_ARB_DIS (PCI_CFG[1] pin value)	Determines if the PCI bridge is the PCI arbiter on the PCI bus. Set or cleared by the PIC_CFG[1] pin value after hard reset. 0 PCI bridge is the PCI arbiter. 1 PCI bridge is not the PCI arbiter. The PCI bridge presents its **request** on $\overline{\text{REQ0}}$ to the external arbiter and receives its **grant** on $\overline{\text{GNT0}}$.
14	Parking mode	Controls which device receives the bus grant when there are no outstanding bus requests and the bus is idle 0 The bus is parked with the last device to use the bus. 1 The bus is parked with the PCI bridge.
13–7	—	Reserved, should be cleared
6–4	PCI bus master priorities	Determines the arbitration priority given to the different masters on the PCI bus. Bit 6 corresponds to the priority of the master sourcing $\overline{\text{REQ0}}$, bit 5 corresponds to $\overline{\text{REQ1}}$, and bit 4 corresponds to $\overline{\text{REQ2}}$. 0 Master *n* has a low priority. 1 Master *n* has a high priority.
3–1	—	Reserved, should be cleared
0	PCI bridge master priority	Determines the PCI bridge's arbitration priority 0 The PCI bridge has a low priority. 1 The PCI bridge has a high priority.

9.11.2.24 PCI Hot Swap Register Block

The PCI Hot Swap register block, shown in Figure 9-56, is a set of registers in a capability structure. It contains the Hot Swap control status register itself, as well as other fields as required by the capabilities list format.

	31–24	23–16
Field	—	HS_CSR
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x4A	

	15–8	7–0
Field	NXT_PTR	CAP_ID
Reset	0000_0000_0000_0110	
R/W	R/W	R
Addr	0x48	

Figure 9-56. Hot Swap Register Block

Table 9-43 describes the Hot Swap register block fields.

Table 9-43. Hot Swap Register Block Field Descriptions

Bits	Name	Description
31–24	—	Reserved. Should be cleared.
23–16	HS_CSR	Hot Swap control status register; see Section 9.11.2.25, “PCI Hot Swap Control Status Register.”
15–8	NXT_PTR	Next pointer—an offset into the device’s PCI configuration space for the location of the next item in the capabilities linked list. A value of 0x00 indicates that this is the last item in the list.
7–0	CAP_ID	CompactPCI ® Hot Swap capability ID (read only).

9.11.2.25 PCI Hot Swap Control Status Register

Figure 9-57 and Table 9-44 describe the Hot Swap control status register.

	23	22	21	20	19	18	17	16
Field	INS	EXT	—		LOO	—	EIM	—
Reset	0000_0000							
R/W	R/W							
Addr	0x4A							

Figure 9-57. Hot Swap Control Status Register

Table 9-44. Hot Swap Control Status Register Field Descriptions

Bit	Name	Description
23	INS	$\overline{\text{ENUM}}$ status: insertion. Write a ‘1’ to clear this bit. 0 $\overline{\text{ENUM}}$ is not asserted. 1 $\overline{\text{ENUM}}$ is asserted.
22	EXT	$\overline{\text{ENUM}}$ status: extraction. Write a ‘1’ to clear this bit. 0 $\overline{\text{ENUM}}$ is not asserted. 1 $\overline{\text{ENUM}}$ is asserted.
21–20	—	Reserved. Should be cleared.
19	LOO	LED on/off when the hardware is in state H2. Read/write-able. 0 LED off 1 LED on
18	—	Reserved. Should be cleared.
17	EIM	$\overline{\text{ENUM}}$ signal mask. Read/write-able. 0 Enable signal 1 Mask signal
16	—	Reserved. Should be cleared.

9.11.2.26 PCI Configuration Register Access from the Core

The 60x bus master cannot directly access the PCI configuration registers because they are not in the internal memory-mapped configuration register’s space. The 60x bus master must first load CFG_ADDR (at offset 0x10900 in the memory-mapped configuration registers block) with a 32-bit register address in the form ‘0x8000_0*nnn*,’ where *nnn* is the address offset of the desired PCI configuration register. The data can then be accessed in CFG_DATA (at offset 0x10904 in the internal memory map). See Section 9.9.1.4.4, “Host Mode Configuration Access.”

When accessing the PCI bridge’s PCI configuration registers with the 60x bus master, note the following:

- The bus number and device number fields of the CFG_ADDR register should be cleared.
- Accesses to CFG_ADDR or CFG_DATA that are greater than 4 bytes generate an illegal register access error setting ECR[IRA]; see Section 9.11.1.11, “Error Control Register (ECR).”
- Accesses to CFG_DATA without a valid offset in CFG_ADDR generates an I/O transaction on the PCI bus.

9.11.2.27 PCI Configuration Register Access in Big-Endian Mode

Since the local CPU (internal core or external) is operating in big-endian mode, software must byte-swap the data of the configuration register before performing an access. That is, the data appears in the core register in ascending significance byte order (LSB to MSB). Software loads the configuration register address and the configuration register data into the core register in ascending significance byte order (LSB to MSB).

Note that in the following examples, the data in the configuration register (at 0x18) is shown in little-endian order. This is because all the internal registers are intrinsically little endian.

EXAMPLE: configuration sequence, 2-byte data write to register at address offset 0x1A for PCI bus.

```
Initial values:
         r0 contains 0x1800_0080
         r1 contains IMMR+0x10900
         r2 contains IMMR+0x10904
         r3 contains 0xDDCC_BBAA
         Register at 0x18 contains 0xFFFF_FFFF (1B to 18)
Code sequence:
         stw      r0,0(r1)
         sth      r3,2(r2)
Results: Address IMMR+0x10900 contains 0x8000_0018 (MSB to LSB)
         Address IMMR+0x10904 contains 0xXXXX_AABB (MSB to LSB) where 'XXXX' is
         the old value and is not affected the sth.
         Note: the address of PCI_CFG_DATA must match the offset address 0x1A.
         Register at 0x18 contains 0xAABB_FFFF (1B to 18)
```

This example shows an address of IMMR+0x10906 used to access the PCI_CFG_DATA. This was done in order to align the data with the address 0x1A. The address used to access PCI_CFG_DATA can have a value of IMMR+0x10904, IMMR+0x10905, IMMR+0x10906, or IMMR+0x10907. The two least-significant bits of the address used to access PCI_CFG_DATA should match the byte-wise offset of the register being accessed. For instance, if 0x0D is the offset of the register being accessed, the address used to access PCI_CFG_DATA must be IMMR+0x10905.

9.11.2.27.1 Additional Information on Endianness

The endianness of both the MPC826x's peripheral logic (GPCR[LE_MODE]—see the following section) and the MPC826x's 603e CPU core (MSR[LE]) must be set to the same endianness configuration—that is both must be set for little- or big-endian operation.

For applications where little-endian (LE) devices, such as those commonly found on the PCI bus, share memory with the MPC826x, it is recommended to leave the MPC826x's 603e core CPU and peripheral logic in the big-endian (BE) modes and then to use a region of the MPC8272 local memory for LE-formatted data. When a little-endian PCI device stores data to this memory region, the MPC8272 internal peripheral logic (in big-endian mode) stores the data into memory in LE format. Likewise, when a little-endian PCI device reads data from this memory region, the MPC826x internal peripheral logic (in BE mode) provides the data to the PCI device in LE format.

A little-endian PCI device can share this LE memory region with the MPC8272 local processor (603e core CPU) running in big endian if, when the MPC8272 accesses that LE region, it uses the **lwbrx** and **stwbrx** commands. The lwbrx command byte-swaps the LE data from that region so the 603e CPU sees the data in BE format. Similarly, the stwbrx command byte-swaps the BE data from the 603e processor being stored to that region of memory, so it is stored into the memory region in LE format.

For the MPC603e and the MPC8272 implementations, there is NO latency difference associated with **lwbrx** and **stwbrx** commands compared to the other load and store commands.

9.11.2.27.2 Notes on GPCR[LE_MODE]

GPCR[LE_MODE] (refer to Section 9.11.1.7, "General Purpose Control Register (GPCR)") determines the endianness of the PCI section of MPC8272. The default value of GPCR[LE_MODE] (offset: 0x1087C) is 0. If LE_MODE is set while a program is executing, care should be taken as to how subsequent accesses to the PCI memory-mapped registers are made. Consider the following two examples (assume internal memory starts at 0x04700000):

EXAMPLE 1— Accessing PCI memory-mapped registers before GPCR[LE_MODE] is set. Assume that one wants to use CPU software to set CTM of PCI DMA0 mode register (DMAMR0[CTM]) located at 0x04710500. The value constructed from the bit field description of the DMAMR0 is 0x00000004. However, the value written to this register is 0x04000000—the byte-swapped version of 0x00000004.

EXAMPLE 2—Accessing PCI memory-mapped registers after GPCR[LE_MODE] is set. Assume that, after GPCR[LE_MODE] is set, one wants to use CPU software to set DMAMR0[CTM]. Because of address munging, this register is now located at 0x04710504. This new address is derived from the following:

1.The register is located at 0x04710500.

2.For a 4-byte access, address munging dictates that the XOR value is 0b100 (refer to Chapter 4 of the *Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture*).

3.The last three bits of 0x04710500 is 0b000.

4.XOR 0b000 with 0b100 (0b000 ⊕ 0b100 = 0b100).

5.Therefore, the munged address of this register would be 0x04710504.

Therefore, to set CTM in PCI DMA0 mode register, 0x00000004 is written to 0x04710504.

9.11.2.28 Initializing the PCI Configuration Registers

The configuration registers are initialized to the reset values shown in the register descriptions. However, they can also be initialized to user-defined values loaded directly from the EEPROM used to configure the MPC8272 by setting the ALD_EN (auto-load enable) bit in the hard reset configuration word; refer to Section 5.4.1, “Hard Reset Configuration Word.”

To initialize configuration registers from an EEPROM, the user builds a contiguous table of register initialization data structures in a user-defined space within the EEPROM. Each data structure, shown in Figure 9-58, contains the address of a specific register and its initialization data, as well as some control information. The last data structure entry in the table is marked by setting its ‘Last’ bit.

Offset From the Table Start Address	0	28	29	30	31
0x00	Destination Address				
0x04	—		Last	Size[1–0]	
0x08	Destination Data				
0x0C	⋮				

Figure 9-58. Data Structure for Register Initialization

Table 9-45 describes the data structure fields.

Table 9-45. Bit Settings for Register Initialization Data Structure

Offset	Bits	Name	Description
0x00	0–31	Address	Contains the absolute destination address to which the data is written
0x04	0–28	—	Reserved, should be cleared
	29	Last	Indicates that this is the last initialization transaction to be performed 0 Not last transaction 1 Last transaction
	30–31	Size	Data size in bytes 00 4 bytes 01 1 byte 10 2 bytes 11 3 bytes
0x08	0–31	Data	Contains the data to be written to the specified address. Data bytes are written according to the value specified in the Size field and according to big-endian byte ordering.

Note that the data structure description assumes the following:

- Addresses refer to 60x bus addresses.
- Address and data byte ordering are big endian.
- Accesses to PCI configuration registers are indirect (through PCI CFG_ADDR and PCI CFG_DATA).

A pointer located at address 0x4 of the EEPROM (right after the hard reset configuration word) defines the beginning of the initialization table. The table should be placed beyond the reset configuration data to avoid the EEPROM bytes dedicated to the 8 possible hard reset configuration words (refer to Section 5.4.1, “Hard Reset Configuration Word,” and Figure 9-59).

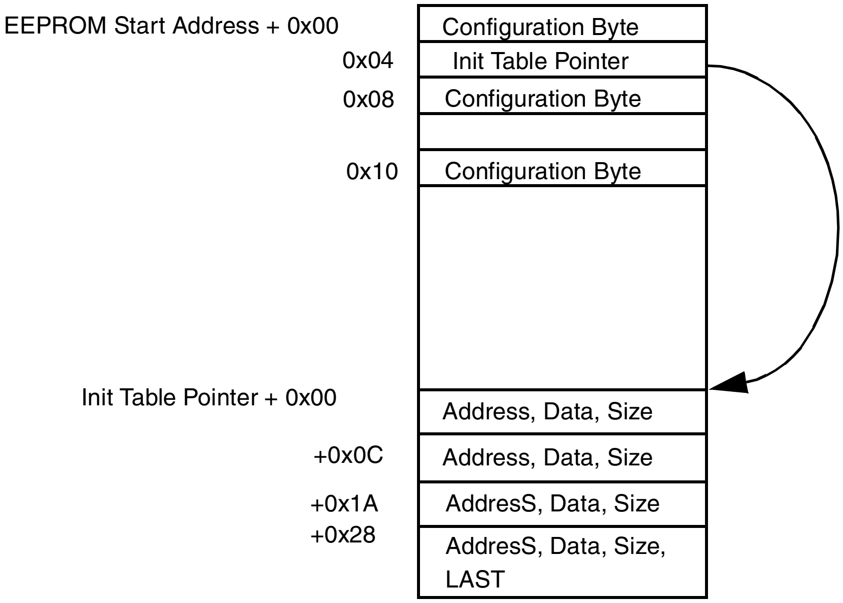


Figure 9-59. PCI Configuration Data Structure for the EEPROM

After a hard reset, if the auto-load enable bit has been set in the hard reset configuration word, a special internal CP routine checks the EEPROM contents and loads the configuration data into the specified addresses. Note that the initialization data can be loaded into any memory location (not restricted to the PCI configuration space) by this routine.

9.12 Message Unit (I_2O)

The 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 processor(s) 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. The PCI bridge provides a messaging unit to further facilitate communications between host and peripheral. The PCI bridge's message unit can operate with either generic messages and door bell registers, or as an I_2O interface.

9.12.1 Message Registers

The PCI bridge contains two inbound message registers and two outbound message registers. The registers are each 32 bits. The inbound registers allow a remote host or PCI master to write a 32-bit value, which in turn causes an interrupt to the local processor that implements the PowerPC architecture because the register indirectly drives an interrupt line to the local processor. The outbound register allows the local processor to write an outbound message which, in turn, causes the outbound interrupt signal $\overline{\text{INTA}}$ to assert.

The interrupt to the local processor is cleared by setting the appropriate bit in the inbound message interrupt status register. The interrupt to PCI ($\overline{\text{INTA}}$) is cleared by setting the appropriate bit in the outbound interrupt status register.

9.12.1.1 Inbound Message Registers (IMR*x*)

The inbound message registers, described in Figure 9-60 and Figure 9-46, are accessible from the PCI bus and the 60x bus in both host and agent modes.

	31 … 16
Field	IMSG*x*
Reset	Undefined
R/W	R/W
Addr	0x10452 (IMR0); 0x10456 (IMR1)

	15 … 0
Field	IMSG*x*
Reset	Undefined
R/W	R/W
Addr	0x10450 (IMR0); 0x10454 (IMR1)

Figure 9-60. Inbound Message Registers (IMR*x*)

Table 9-46. IMR*x* Field Descriptions

Bits	Name	Description
31–0	IMSG*x*	Inbound message *x*. Contains generic data to be passed between the local processor and external hosts.

9.12.1.2 Outbound Message Registers (OMR*x*)

The outbound message registers, described in Figure 9-61 and Figure 9-47, are accessible from the PCI bus and the 60x bus in both host and agent modes.

	31	16
Field	OMSG*x*	
Reset	Undefined	
R/W	R/W	
Addr	0x1045A (OMR0); 0x1045E (OMR1)	

	15	0
Field	OMSG*x*	
Reset	Undefined	
R/W	R/W	
Addr	0x10458 (OMR0); 0x1045C (OMR1)	

Figure 9-61. Outbound Message Registers (OMR*x*)

Table 9-47. OMR*x* Field Descriptions

Bits	Name	Description
31–0	OMSG*x*	Outbound message *x*. Contains generic data to be passed between the local processor and external hosts.

9.12.2 Door Bell Registers

The PCI bridge contains an inbound and an outbound door bell register. The registers are 32 bits. The inbound door bell allows a remote processor to set a bit in the register from the PCI bus. This, in turn, causes the PCI bridge to generate an interrupt to the local processor. The local processor can write to the outbound register that causes the outbound interrupt signal $\overline{\text{INTA}}$ to assert thus interrupting the remote processor on the PCI bus.

9.12.2.1 Outbound Doorbell Register (ODR)

ODR, described in Figure 9-62 and Table 9-48, is accessible from the PCI bus and the 60x bus in both host and agent modes.

	31 29	28 16
Field	—	ODR*x*
Reset	0000_0000_0000_0000	
R/W	Refer to Table 9-48.	
Addr	0x10462	

	15 0
Field	ODR*x*
Reset	0000_0000_0000_0000
R/W	Refer to Table 9-48.
Addr	0x10460

Figure 9-62. Outbound Doorbell Register (ODR)

Table 9-48. ODR Field Descriptions

Bits	Name	Access	Description
31–29	—	R	Reserved, should be cleared
28–0	ODR*x*	Write 1 to set from local processor. Write 1 to clear from PCI.	Outbound door bell *x*, where *x* is each bit. Writing a bit in this register from the local processor causes an interrupt ($\overline{\text{INTA}}$) to be generated.

9.12.2.2 Inbound Doorbell Register (IDR)

IDR, described in Figure 9-63 and Table 9-49, is accessible from the PCI bus and the 60x bus in both host and agent modes.

	31	30	16
Field	IMC	IDR*x*	
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x1046A		

	15	0
Field	IDR*x*	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10468	

Figure 9-63. Inbound Doorbell Register (IDR)

Table 9-49. IDR Field Descriptions

Bits	Name	Access	Description
31	IMC	Write 1 to set from PCI. Write 1 to clear from local processor.	Machine check. Writing to this bit will generate a machine check interrupt to the local processor.
30–0	IDR*x*	Write 1 to set from PCI. Write 1 to clear from local processor.	Inbound door bell *x*, where *x* is each bit. Writing a bit in this register from the PCI bus causes an interrupt to be generated through the PCI bridge to the local processor.

9.12.3 I_2O Unit

The Intelligent Input Output specification (I_2O) was established in the industry to allow architecture-independent I/O subsystems to communicate with an OS through an abstraction layer. The specification is centered around a message passing scheme. An I_2O embedded peripheral (IOP) is comprised of memory, processor, and input/output device(s). An IOP dedicates space in its local memory to hold inbound (from the remote host) and outbound (to the remote host) messages. The space is managed as memory-mapped FIFOs, with pointers to this memory maintained in hardware.

Messages are made up of frames that are a minimum of 64 bytes in length. The message frame address (MFA) is the address that points to the first byte of the message frame. The messages are located in local-system memory. Tracking of the status and location of these messages is done with four FIFOs (two FIFOs for inbound and two for outbound messages) also located in local-system memory. Hardware registers inside the PCI bridge's core logic manage these FIFOs. One FIFO in each queue keeps track of the free MFAs (Free_LIST FIFO). The other FIFO keeps track of the MFAs that have posted messages (Post_LIST FIFO). Figure 9-64 shows an example of the message queues, although there is no specific order that these queues must follow.

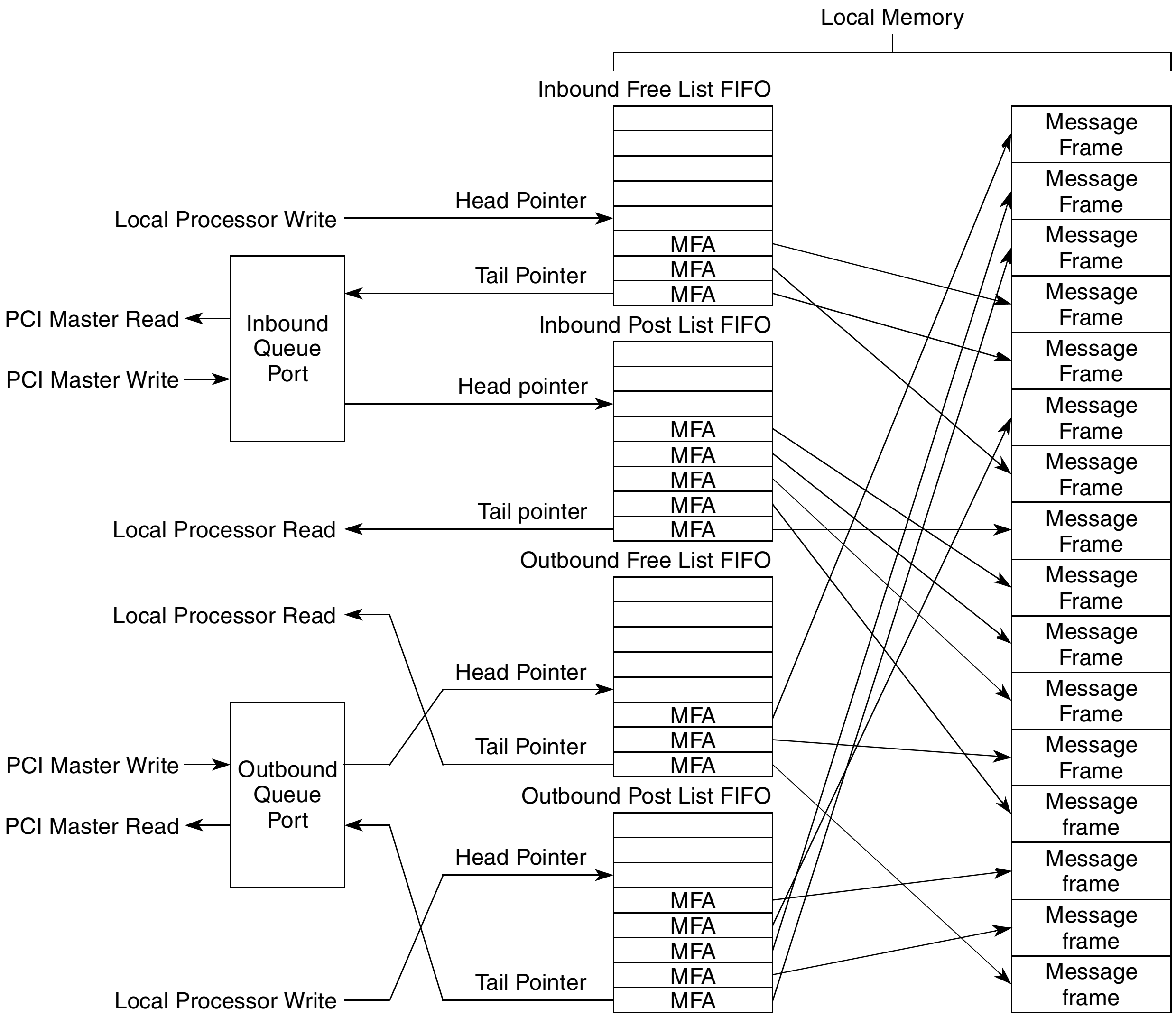


Figure 9-64. I_2O Message Queue

I_2O defines extensions for the PCI bus hardware through which message queues are managed in hardware.

9.12.3.1 PCI Configuration Identification

A host identifies an IOP by its PCI class code. When I_2O is enabled, configuration information is provided through the PCI configuration space to the host. Refer to the following:

- Section 9.11.2.6, “PCI Bus Programming Interface Register”
- Section 9.11.2.7, “Subclass Code Register”
- Section 9.11.2.8, “PCI Bus Base Class Code Register”

9.12.3.2 Inbound FIFOs

The inbound FIFO allows external PCI masters to post messages to the local processor. I_2O defines two inbound FIFOs—an inbound post FIFO and an inbound free FIFO.

The following registers should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

9.12.3.2.1 Inbound Free_FIFO Head Pointer Register (IFHPR) and Inbound Free_FIFO Tail Pointer Register (IFTPR)

The inbound free list FIFO holds the list of empty inbound MFAs. The external PCI master reads IFQPR (refer to Section 9.12.3.4.1, “Inbound FIFO Queue Port Register (IFQPR)”), which returns the MFA pointed to by the inbound free list tail pointer register, (IFTPR+QBAR). The PCI bridge’s I_2O unit then advances IFTPR.

If the inbound free list is empty (no free MFA entries), the unit returns 0xFFFF_FFFF.

Free MFAs from the local processor are posted to the inbound free list FIFO that is pointed to by the inbound free_FIFO head pointer register, described in Figure 9-65 and Table 9-50. The local processor is responsible for updating this register.

	31 20	19 16
Field	QBA	IFHP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104A2	

	15 2	1 0
Field	IFHP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104A0	

Figure 9-65. Inbound Free_FIFO Head Pointer Register (IFHPR)

Table 9-50. IFHPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	IFHP	Inbound free_fifo head pointer. Local memory offset of the head pointer of the inbound free list FIFO.
1–0	—	Reserved, should be cleared

Free MFAs are picked up by the PCI masters that are pointed to by the inbound free_FIFO tail pointer, described in Figure 9-66 and Table 9-51. The PCI read is performed at the inbound queue port. Hardware automatically advances this register after every read.

	31 … 20	19 … 16
Field	QBA	IFTP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104AA	

	15 … 2	1 … 0
Field	IFTP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104A8	

Figure 9-66. Inbound Free_FIFO Tail Pointer Register (IFTPR)

Table 9-51. IFTPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	IFTP	Inbound free_FIFO tail pointer. Local memory offset of the tail pointer of the inbound free list FIFO.
1–0	—	Reserved, should be cleared

9.12.3.2.2 Inbound Post_FIFO Head Pointer Register (IPHPR) and Inbound Post_FIFO Tail Pointer Register (IPTPR)

The inbound post FIFO holds MFAs from external PCI masters that are posted to the local processor. PCI masters, external to the PCI bridge, write to the head of the FIFO by writing the MFA to IFQPR (refer to Section 9.12.3.4.1, “Inbound FIFO Queue Port Register (IFQPR)”). The I_2O unit transfers the MFA to the location pointed to by the IPHPR. The local address is QBAR + IPHPR.

Once the MFA has been written to the queue in local memory, the PCI bridge’s I_2O unit advances the IPHPR to set up for the next message. This causes an interrupt to be asserted to the local processor. The inbound post queue interrupt bit in the inbound interrupt status register (IMISR[IPQI]) is set to indicate this condition (refer to Table 9-62). The local processor acknowledges the message (that is, MFA) by writing a one to the appropriate status bit (IMISR[IPQI]) to clear it. The local processor fetches the MFA by reading the contents of the IPTPR. After the local processor has read the message pointed to by the MFA, the local processor must advance the IPTPR. Once the processor has completed use of the message, it must return the message buffer (that is, MFA) to the inbound free list FIFO.

PCI masters post MFAs to the inbound post list FIFO that is pointed to by the inbound post_FIFO head pointer register, described in Figure 9-67 and Table 9-52. The PCI writes are addressed to the inbound queue port. Hardware (in the I_2O module) automatically advances the IPHPR after every write.

	31 20	19 16
Field	QBA	IPHP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104B2	

	15 2	1 0
Field	IPHP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104B0	

Figure 9-67. Inbound Post_FIFO Head Pointer Register (IPHPR)

Table 9-52. IPHPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	IPHP	Inbound post_FIFO head pointer. Local memory offset of the head pointer of the inbound post list FIFO.
1–0	—	Reserved, should be cleared

MFAs posted by PCI hosts are picked up by the local processor through the inbound post_FIFO tail pointer register, described in Figure 9-68 and Table 9-53. The local processor is responsible for updating this register.

	31 20	19 16
Field	QBA	IPTP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104BA	

	15 2	1 0
Field	IPTP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104B8	

Figure 9-68. Inbound Post_FIFO Tail Pointer Register (IPTPR)

Table 9-53. IPTPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	IPTP	Inbound post_FIFO tail pointer. Local memory offset of the tail pointer of the inbound post list FIFO.
1–0	—	Reserved, should be cleared

9.12.3.3 Outbound FIFOs

The outbound queues are used to send messages from the local processor to a remote host processor. I_2O defines two outbound FIFOs—an outbound post FIFO and an outbound free FIFO.

The following registers should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

9.12.3.3.1 Outbound Free_FIFO Head Pointer Register (OFHPR) and Outbound Free_FIFO Tail Pointer Register (OFTPR)

The outbound free list FIFO holds the MFAs of the empty outbound message locations in local memory. When the local processor is ready to send an outbound message, it first fetches an empty MFA by reading the OFTPR. It then writes the message into the MFA. The OFTPR is managed by the local processor.

When an external PCI master has completed use of a message that was posted in the outbound post FIFO and wants to return the MFA to the free list, it writes to OFQPR (refer to Section 9.12.3.4.2, “Outbound FIFO Queue Port Register (OFQPR)”). The PCI bridge’s I_2O unit then writes the MFA to the OFHPR. This, in turn, causes the outbound free head pointer to be advanced.

Free MFAs are returned by the PCI masters to the outbound free list FIFO that is pointed to by the outbound free_FIFO head pointer register, described in Figure 9-69 and Table 9-54. The PCI write references the outbound queue port. The I_2O hardware automatically advances the address, (that is, OFHPR) after every write.

	31 20	19 16
Field	QBA	OFHP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104C2	

	15 2	1 0
Field	OFHP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104C0	

Figure 9-69. Outbound Free_FIFO Head Pointer Register (OFHPR)

Table 9-54. OFHPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR.
19–2	OFHP	Outbound free_FIFO head pointer. Local memory offset of the head pointer of the outbound free list FIFO.
1–0	—	Reserved, should be cleared

Free MFAs are picked up by the local processor pointed to by the outbound free_FIFO tail pointer register, described in Figure 9-70 and Table 9-55. This register is updated by the local processor.

	31 20	19 16
Field	QBA	OFTP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104CA	

	15 2	1 0
Field	OFTP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104C8	

Figure 9-70. Outbound Free_FIFO Tail Pointer Register (OFTPR)

Table 9-55. OFTPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	OFTP	Outbound free_FIFO tail pointer. Local memory offset of the tail pointer of the outbound free list FIFO.
1–0	—	Reserved, should be cleared

9.12.3.3.2 Outbound Post_FIFO Head Pointer Register (OPHPR) and Outbound Post_FIFO Tail Pointer Register (OPTPR)

The outbound post FIFO holds MFAs that are posted from the local processor to external processors. The local processor places messages in the outbound post FIFO by writing to the MFA to OPHPR + QBAR. The local processor must then advance the OPHPR.

The PCI bridge's PCI interrupt is generated ($\overline{\text{INTA}}$) when the FIFO is not empty (head and tail pointers are not equal). The outbound post queue interrupt bit is set in the outbound interrupt status register. The status bit is cleared when the head and tail pointers are equal. The interrupt can be masked using the outbound interrupt mask register.

An external PCI master reads the outbound queue port register. This causes the PCI bridge's I_2O unit to read the MFA from local memory pointed to by the OPTPR + QBAR. The unit then advances the OPTPR.

When the FIFO is empty (head and tail pointers are equal), the unit returns 0xFFFF_FFFF.

The local processor posts MFAs to the outbound post list FIFO that is pointed to by the outbound post_FIFO head pointer register, described in Figure 9-71 and Table 9-56. The local processor is responsible for updating this register.

	31	20	19	16
Field	QBA		OPHP	
Reset	0000_0000_0000_0000			
R/W	R		R/W	
Addr	0x104D2			

	15	2	1	0
Field	OPHP		—	
Reset	0000_0000_0000_0000			
R/W	R/W		R	
Addr	0x104D0			

Figure 9-71. Outbound Post_FIFO Head Pointer Register (OPHPR)

Table 9-56. OPHPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	OPHP	Outbound post_FIFO head pointer. Local memory offset of the head pointer of the outbound post list FIFO.
1–0	—	Reserved, should be cleared

Posted MFAs are picked up by PCI hosts that are pointed to by the outbound post_FIFO tail pointer register, described in Figure 9-72 and Table 9-57. The PCI read is performed at the outbound queue port. Hardware automatically advances this register after every read.

	31 20	19 16
Field	QBA	OPTP
Reset	0000_0000_0000_0000	
R/W	R	R/W
Addr	0x104DA	

	15 2	1 0
Field	OPTP	—
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x104D8	

Figure 9-72. Outbound Post_FIFO Tail Pointer Register (OPTPR)

Table 9-57. OPTPR Field Descriptions

Bits	Name	Description
31–20	QBA	Queue base address. When read returns the contents of QBAR bits 31–20.
19–2	OPTP	Outbound post_FIFO tail pointer. Local memory offset of the tail pointer of the outbound post list FIFO.
1–0	—	Reserved, should be cleared

9.12.3.4 I$_2$O Registers

9.12.3.4.1 Inbound FIFO Queue Port Register (IFQPR)

IFQPR is used by PCI masters to access inbound messages in local memory. The local processor does not have access to this port. IFQPR should be accessed only from the PCI bus. IFQPR is described in Figure 9-73 and Table 9-58.

	31 16
Field	IFQP
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10442

	15 0
Field	IFQP
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10440

Figure 9-73. Inbound FIFO Queue Port Register (IFQPR)

Table 9-58. IFQPR Field Descriptions

Bits	Name	Description
31–0	IFQP	Inbound FIFO queue port. Reading this register will return the MFA from inbound free list FIFO. Writing to this register will post the MFA to the inbound post list FIFO.

9.12.3.4.2 Outbound FIFO Queue Port Register (OFQPR)

OFQPR is used by PCI masters to access outbound messages in local memory. Local processor does not have access to this port. OFQPR should be accessed only from the PCI bus. OFQPR is described in Figure 9-74 and Table 9-59.

	31 … 16
Field	OFQP
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10446

	15 … 0
Field	OFQP
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10444

Figure 9-74. Outbound FIFO Queue Port Register (OFQPR)

Table 9-59. OFQPR Field Descriptions

Bits	Name	Description
31–0	OFQP	Outbound FIFO queue port. Reading this register will return the MFA from outbound post list FIFO. Writing this register will post the MFA to the outbound free list FIFO.

9.12.3.4.3 Outbound Message Interrupt Status Register (OMISR)

OMISR contains the interrupt status of the I_2O, door bell, 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]. This 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 should be accessed only from the PCI bus IFQPR should be accessed only from the PCI bus.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	Refer to Table 9-60.
Addr	0x10432

	15–6	5	4	3	2	1	0
Field	—	OPQI	—	ODI	—	OM1I	OM0I
Reset	0000_0000_0000_0000						
R/W	Refer to Table 9-60.						
Addr	0x10430						

Figure 9-75. Outbound Message Interrupt Status Register (OMISR)

Table 9-60 describes OMISR fields.

Table 9-60. OMISR Field Descriptions

Bits	Name	R/W	Description
31–6	—	R	Reserved, should be cleared
5	OPQI	R	Outbound post queue interrupt. When set indicates that a message or messages are posted in the outbound queue. To clear the interrupt, software has to read all MFAs in the outbound post FIFO. This bit is set regardless of the state of the OPQIM mask bit.[1]
4	—	R	Reserved, should be cleared
3	ODI	R	Outbound doorbell interrupt. When set indicates that there is an outbound doorbell interrupt.
2	—	R	Reserved, should be cleared
1	OM1I	Read/ Write 1 to clear	Outbound message 1 interrupt. When set indicates that there is an Outbound message 1 interrupt.
0	OM0I	Read/ Write 1 to clear	Outbound message 0 interrupt. When set indicates that there is an Outbound message 0 interrupt

[1] Note that when conditions for the outbound post queue interrupt assertion are valid, and OMIMR[OPQIM] is set, OMISR[OPQI] is cleared. The application should always clear OMIMR[OPQIM] before referring to the content of OMISR[OPQI].

9.12.3.4.4 Outbound Message Interrupt Mask Register (OMIMR)

OMIMR contains the interrupt mask of the I_2O, door bell, and message register events generated by the local processor. OMIMR should be accessed only from the PCI bus.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	Refer to Table 9-61.
Addr	0x10436

	15–6	5	4	3	2	1	0
Field	—	OPQIM	—	ODIM	—	OM1IM	OM0IM
Reset	0000_0000_0000_0000						
R/W	Refer to Table 9-61.						
Addr	0x10434						

Figure 9-76. Outbound Message Interrupt Mask Register (OMIMR)

Table 9-61 describes OMIMR fields.

Table 9-61. OMIMR Field Descriptions

Bits	Name	R/W	Description
31–6	—	R	Reserved, should be cleared
5	OPQIM	RW	Outbound post queue interrupt mask 0 Outbound post queue interrupt is allowed. 1 Outbound post queue interrupt is masked.
4	—	R	Reserved, should be cleared
3	ODIM	RW	Outbound doorbell interrupt mask 0 Outbound doorbell interrupt is allowed. 1 Outbound doorbell interrupt is masked.
2	—	R	Reserved, should be cleared
1	OM1IM	RW	Outbound message 1 interrupt mask 0 Outbound message 1 interrupt is allowed. 1 Outbound message 1 interrupt is masked.
0	OM0IM	RW	Outbound message 0 interrupt mask 0 Outbound message 0 interrupt is allowed. 1 Outbound message 0 interrupt is masked.

9.12.3.4.5 Inbound Message Interrupt Status Register (IMISR)

This register contains the interrupt status of the I_2O, door bell, and message register events. Writing a 1 to the corresponding set bit will clear the bit. The events are generated by the PCI masters. IMISR should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

	31	16
Field	—	
Reset	0000_0000_0000_0000	
R/W	Refer to Table 9-62.	
Addr	0x10482	

	15	9	8	7	6	5	4	3	2	1	0
Field	—		OFOI	IPOI	—	IPQI	MCI	IDI	—	IM1I	IM0I
Reset	0000_0000_0000_0000										
R/W	Refer to Table 9-62.										
Addr	0x10480										

Figure 9-77. Inbound Message Interrupt Status Register (IMISR)

Table 9-62 describes IMISR fields.

Table 9-62. IMISR Field Descriptions

Bits	Name	Access	Description
31–9	—	R	Reserved, should be cleared.
8	OFOI	R/Write 1 to clear	Outbound Free Overflow Interrupt. When set indicates that the Outbound Free_FIFO Head pointer is equal to the Outbound Free_FIFO Tail pointer and the queue is full. A machine check interrupt is generated.
7	IPOI	R/Write 1 to clear	Inbound Post Overflow Interrupt. When set indicates that the Inbound Post_FIFO Head pointer is equal to the Inbound Post_FIFO Tail pointer and the queue is full. A machine check interrupt is generated.
6	—	R	Reserved, should be cleared.
5	IPQI	R/Write 1 to clear	Inbound Post Queue Interrupt. When set indicates that the PCI master has posted an MFA to the Inbound Post queue.
4	MCI	R	Machine check interrupt. When set indicates that a machine check interrupt condition has been generated by setting the Inbound doorbell register’s bit 31. The interrupt is cleared by resetting the Inbound doorbell register’s bit 31.
3	IDI	R	Inbound doorbell interrupt. When set indicates that there is an Inbound Doorbell interrupt.
2	—	R	Reserved, should be cleared.
1	IM1I	R/Write 1 to clear	Inbound message 1 interrupt. When set indicates that there is an Inbound message 1 interrupt.
0	IM0I	R/Write 1 to clear	Inbound message 0 interrupt. When set indicates that there is an Inbound message 0 interrupt.

9.12.3.4.6 Inbound Message Interrupt Mask Register (IMIMR)

This register contains the interrupt mask of the I_2O, door bell, and message register events generated by the PCI master. IMIMR should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

	31	16
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10486	

	15	9	8	7	6	5	4	3	2	1	0
Field	—		OFOIM	IPOIM	—	IPQIM	MCIM	IDIM	—	IM1IM	IM0IM
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x10484										

Figure 9-78. Inbound Message Interrupt Mask Register (IMIMR)

Table 9-63 describes IMIMR fields.

Table 9-63. IMIMR Field Descriptions

Bits	Name	Description
31–9	—	Reserved, should be cleared.
8	OFOIM	Outbound free overflow interrupt mask 0 Outbound free overflow interrupt is allowed. 1 Outbound free overflow interrupt is masked.
7	IPOIM	Inbound post overflow interrupt mask 0 Inbound post overflow interrupt is allowed. 1 Inbound post overflow interrupt is masked.
6	—	Reserved, should be cleared.
5	IPQIM	Inbound post queue interrupt mask 0 Inbound post queue interrupt is allowed. 1 Inbound post queue interrupt is masked.
4	MCIM	Machine check interrupt mask 0 Machine check interrupt from the inbound doorbell register is allowed. 1 Machine check interrupt is masked.
3	IDIM	Inbound doorbell interrupt mask 0 Inbound doorbell interrupt is allowed. 1 Inbound doorbell interrupt is masked.
2	—	Reserved, should be cleared.

Table 9-63. IMIMR Field Descriptions (continued)

Bits	Name	Description
1	IM1IM	Inbound message 1 interrupt mask 0 Inbound doorbell interrupt is allowed. 1 Inbound doorbell interrupt is masked.
0	IM0IM	Inbound message 0 interrupt mask 0 Inbound message 0 interrupt is allowed. 1 Inbound message 0 interrupt is masked.

9.12.3.4.7 Messaging Unit Control Register (MUCR)

This register allows software to enable and setup the size of the inbound and outbound FIFOs. MUCR should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

	31–16
Field	—
Reset	0000_0000_0000_0002
R/W	R/W
Addr	0x104E6

	15–6	5–1	0
Field	—	CQS	CQE
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x104E4		

Figure 9-79. Messaging Unit Control Register (MUCR)

Table 9-64 describes MUCR fields.

Table 9-64. MUCR Field Descriptions

Bits	Name	Access	Description
31–6	—	R	Reserved, should be cleared

Table 9-64. MUCR Field Descriptions (continued)

Bits	Name	Access	Description
5–1	CQS	RW	Circular queue size. CQS refers to each individual queue, not the total size of all four queues together. 00001 4K entries (16 Kbytes) 00010 8K entries (32 Kbytes) 00100 16K entries (64 Kbytes) 01000 32K entries (128 Kbytes) 10000 64K entries (256 Kbytes) All others reserved.
0	CQE	RW	Circular queue enable. When set will allow PCI masters to access the inbound and outbound queue ports. Writes are ignored and reads will return 0xFFFF_FFFF when this bit is cleared. Normally, this bit is set only if software has initialized all pointers and configuration registers.

9.12.3.4.8 Queue Base Address Register (QBAR)

This register specifies the beginning of the circular queue structure in local memory. The following QBAR should be accessed only from the 60x bus and only in agent mode. Accesses while in host mode or from the PCI bus have undefined results.

	31 – 20	19 – 16
Field	QBA	—
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x104F2	

	15 – 0
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x104F0

Figure 9-80. Queue Base Address Register (QBAR)

Table 9-65 describes QBAR fields.

Table 9-65. QBAR Field Descriptions

Bits	Name	Access	Description
31–20	QBA	RW	Queue base address. Base address of circular queue in local memory. It must be aligned to a 1-Mbyte boundary.
19–0	—	R	Reserved, should be cleared

9.13 DMA Controller

The PCI bridge's DMA controller transfers blocks of data independent of the local core or PCI hosts. Data movement occurs on the PCI and/or 60x bus. The PCI Bridge's DMA module has four high-speed DMA channels with an aggregate bandwidth conservatively estimated at 210 Mbytes per second, for 60x to PCI transfer. The channels share 144 bytes of DMA-dedicated buffer space to facilitate the gathering and sending of data. Both the local core 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 core and remote PCI masters.
- Unaligned transfer capability
- Data chaining and direct mode
- Interrupt on completed segment, chain, and error
- Support for all transfer combinations between 60x memory and PCI memory: 60x-to-60x, PCI-to-PCI, 60x-to-PCI, and PCI-to-60x

Figure 9-81 shows a block diagram of the DMA controller.

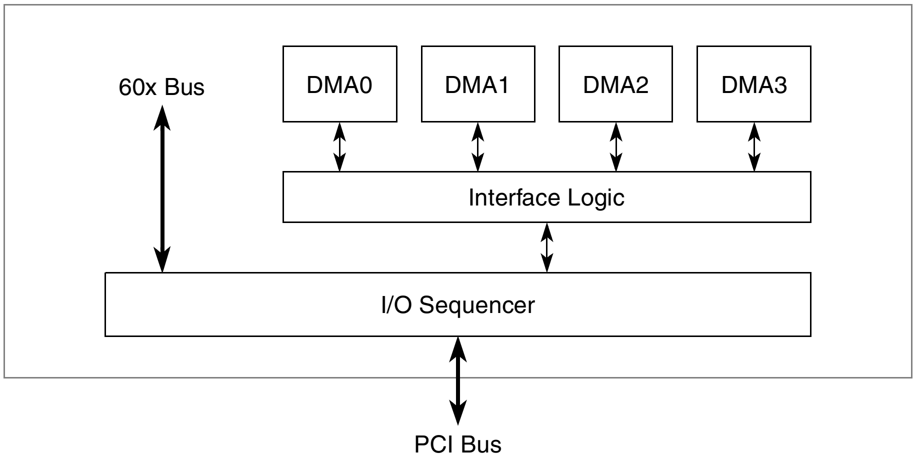


Figure 9-81. DMA Controller Block Diagram

9.13.1 DMA Operation

The DMA controller operates in two modes—chaining and direct. In direct mode, the software is responsible for initializing the source, destination and byte count registers. In chaining mode, the software first must build a chain of descriptor segments in external memory, residing either on the 60x or PCI bus, and then initialize the current descriptor address register to point to the first descriptor segment in the chain. 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 60x memory addresses.

All 60x memory read operations are cache line reads (32 bytes); the DMA controller selects the appropriate/valid data bytes within the cache line when loading its internal queue. Writing to 60x memory depends on the alignment of the destination address and the size of the transfer. The DMA controller writes 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 a transfer size of less than 8 bytes in address hold mode (DAHE or SAHE is set; refer to Section 9.13.1.6.1, “DMA Mode Register [0–3] (DMAMRx),”) precludes cache line writes.

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.

9.13.1.1 DMA 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 or an error condition has occurred. Below are the initialization steps of a DMA transfer in direct mode.

- Poll the CB (channel busy) bit in the status register to make sure the DMA channel is idle (refer to Section 9.13.1.6.2, “DMA Status Register [0–3] (DMASRx)”).
- Initialize the source, destination and byte count register (refer to Section 9.13.1.6.5, “DMA Destination Address Register [0–3] (DMADARx),” and Section 9.13.1.6.6, “DMA Byte Count Register [0–3] (DMABCRx)”).
- Initialize the CTM (channel transfer mode) bit in the mode register (refer to Section 9.13.1.6.1, “DMA Mode Register [0–3] (DMAMRx)”) to indicate direct mode. Other control parameters in the mode register can also be initialized here if necessary.
- First clear then set the CS (channel start) bit in the mode register to start the DMA transfer.

9.13.1.2 DMA 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 or an error condition has occurred. Below are the initialization steps of a DMA transfer in chaining mode.

- Build a chain of descriptor segments in memory. Refer to the Section 9.13.2, “DMA Segment Descriptors,” for more information.
- Poll the CB (channel busy) bit in the status register to make sure the DMA channel is idle.
- Initialize the current descriptor address register to point to the first descriptor in the chain.

- Initialize the CTM (channel transfer mode) bit in the mode register to indicate chaining mode. Other control parameters in the mode register can also be initialized here if necessary.
- First clear then set the CS (channel start) bit in the mode register to start the DMA transfer.

9.13.1.3 DMA Coherency

The four DMA channels are allocated 4 cache lines (128 bytes) of buffer space in the I/O sequencer module 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 core data cache is selectable during DMA transactions. A snoop bit is provided in the current descriptor address register and the next descriptor address register that allows software to control when the cache is snooped. These bits are described in Section 9.13.1.6.3, “DMA Current Descriptor Address Register [0–3] (DMACDARx),” and Section 9.13.1.6.7, “DMA Next Descriptor Address Registers [0–3] (DMANDARx),” respectively.

9.13.1.4 Halt and Error Conditions

DMA transfers are halted either by clearing the CS (channel start) bit in the mode register or when encountering an error condition. In both cases the application software can 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 status register 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.

NOTE: DMA Operation After Bus Error

After any bus error that occurs in the MPC8272 (either 60x or PCI, not necessarily due to DMA operation), the user must reset the system to avoid DMA malfunction.

9.13.1.5 DMA Transfer Types

The DMA controller supports all transfers between 60x memory and PCI memory: 60x-to-60x, PCI-to-PCI, 60x-to-PCI, and PCI-to-60x. All data is temporarily stored in a 144-byte queue prior to transmission. There are four types of DMA transfers:

- PCI-memory-to-PCI-memory transfers—The DMA controller begins by reading data from PCI memory space and storing it in the DMA queue. Once sufficient data is stored in the queue, the DMA controller begins writing data from the queue to PCI memory space beginning at the destination address. The process is repeated until there is no more data to transfer or an error condition has occurred on the PCI bus.

- PCI-memory-to-60x-memory transfers—The DMA controller initiates reads on the PCI bus and stores the data in the DMA queue. Once sufficient data is stored in the queue, a 60x memory write is initiated. The DMA controller stops the transfer either for an error condition on the PCI bus or 60x bus, or when no data is left to transfer. Reading from PCI memory and writing to 60x memory can occur concurrently.
- 60x-memory-to-PCI-memory transfers—The DMA controller initially fetches data from 60x memory into the DMA queue. As soon as the first data arrives into the queue, the DMA engine initiates write transactions to PCI memory. The DMA controller stops the transfer either when there is an error on the PCI bus or 60x bus, or there is no more data left to transfer. Reading from 60x memory and writing to PCI memory can occur concurrently.
- 60x-memory-to-60x-memory transfers—The DMA controller begins reading data from 60x memory and storing it in the DMA queue. Once sufficient data is stored in the queue, the DMA controller begins writing data to 60x memory space beginning at the destination address. The process is repeated until there is no more data to transfer or an error condition has occurred while accessing memory.

9.13.1.6 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. This section describes the format of the DMA support registers.

9.13.1.6.1 DMA Mode Register [0–3] (DMAMR*x*)

The mode register allows software to start the DMA transfer and to control various DMA transfer characteristics.

	31 24	23 21	20	19	18	17 16
Field	—	BWC	DM_SEN	IRQS	—	DAHTS
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x10502 (DMAMR0); 0x10582 (DMAMR1); 0x10602 (DMAMR2); 0x10682 (DMAMR3)					

	15 14	13	12	11 10	9 8	7	6 4	3	2	1	0
Field	SAHTS	DAHE	SAHE	PRC	—	EOTIE	—	TEM	CTM	CC	CS
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x10500 (DMAMR0); 0x10580 (DMAMR1); 0x10600 (DMAMR2); 0x10680 (DMAMR3)										

Figure 9-82. DMA Mode Register [0–3] (DMAMR*x*)

Table 9-66 describes DMAMR*x* fields.

Table 9-66. DMAMR*x* Field Descriptions

Bits	Name	Description
31–24	—	Reserved, should be cleared
23–21	BWC	Bandwidth control. This field 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 DMA Channels. 000 1 cache line 001 2 cache lines 010 4 cache lines 011 8 cache lines 100 16 cache lines
20	DM_SEN	Direct mode snoop enable. When set allows snooping during direct mode DMA transactions.
19	IRQS	Interrupt steer. When set routes all DMA interrupts to PCI bus through $\overline{\text{INTA}}$. When clear routes all DMA interrupts to local core.
18	—	Reserved, should be cleared
17–16	DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction when DAHE is enabled. 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. Indicates the transfer size used for each transaction when SAHE is enabled. 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. When set will allow 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.
12	SAHE	Source address hold enable. When set will allow the DMA controller to hold the source address constant for every transfer. The size used for the transfer is indicated by SAHTS. Note that hardware supports only aligned transfers for this feature.
11–10	PRC	PCI read command. Indicates the types of PCI read command to use. 00 PCI read 01 PCI read line 10 PCI read multiple 11 Reserved
9–8	—	Reserved, should be cleared

Table 9-66. DMAMR*x* Field Descriptions (continued)

Bits	Name	Description
7	EOTIE	End-of-transfer interrupt enable. When set will generate an interrupt at the completion of a DMA transfer. No EOT interrupt is generated if this bit is cleared. 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.
6–4	—	Reserved, should be cleared
3	TEM	Transfer error mask. This bit determines the DMA response in the event of a transfer error. If this bit is set, the DMA will complete the transfer regardless of whether a transfer error occurs (the TE bit is not set). If this bit is clear, the DMA will halt when a transfer error occurs (TE bit is set).
2	CTM	Channel transfer mode 0 Chaining mode. See Section 9.13.1.2, “DMA Chaining Mode.” 1 Direct mode. See Section 9.13.1.1, “DMA Direct Mode.”
1	CC	Channel continue. When this bit is set, the DMA transfer will restart the transferring process starting at the current descriptor address. This bit applies only to chaining mode and is cleared by hardware after every descriptor read.
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, then 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.

9.13.1.6.2 DMA Status Register [0–3] (DMASR*x*)

The status register reports various DMA conditions during and after the DMA transfer. Writing a 1 to a specific set bit clears the bit.

	31–16
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10506(DMASR0); 0x10586 (DMASR1); 0x10606 (DMASR2); 0x10686 (DMASR3)

	15–8	7	6–3	2	1	0
Field	—	TE	—	CB	EOSI	EOCDI
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x10504 (DMASR0); 0x10584(DMASR1); 0x10604 (DMASR2); 0x10684 (DMASR3)					

Figure 9-83. DMA Status Register [0–3] (DMASR*x*)

Table 9-67 describes DMASR*x* fields.

Table 9-67. DMASR*x* Field Descriptions

Bits	Name	Access	Description
31–8	—	RW	Reserved, should be cleared
7	TE	Read/ Write 1 to clear	Transfer error. This bit is set when there is an error condition during the DMA transfer and the TEM bit is cleared.
6–3	—	R	Reserved, should be cleared
2	CB	Read Only	Channel busy. When set indicates that a DMA transfer is currently in progress. This bit will be cleared as a result of any of the three following conditions: (1) an error, (2) a halt, or (3) completion of the DMA transfer.
1	EOSI	Read/ Write 1 to clear	End-of-segment interrupt. After transferring a segment of data, if the EOSIE bit in the current descriptor address register is set, then this bit will be set and an interrupt is generated. Otherwise, no interrupt is generated.
0	EOCDI	Read/ Write 1 to clear	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 will be set and an interrupt is generated. Otherwise, no interrupt is generated.

9.13.1.6.3 DMA Current Descriptor Address Register [0–3] (DMACDAR*x*)

The current descriptor address register 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 next descriptor into DMANDAR, and executes the current transfer. This process continues until encountering a descriptor whose EOTD (end-of-transfer descriptor) bit is set, which will be the last descriptor to be executed.

	31	16
Field	CDA	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x1050A(DMACDR0); 0x1058A (DMACDR1); 0x1060A (DMACDR2); 0x1068A (DMACDR3)	

	15	5	4	3	2	0
Field	CDA		SNEN	EOSIE	—	
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x10508 (DMACDR0); 0x10588 (DMACDR1); 0x10608 (DMACDR2); 0x10688 (DMACDR3)					

Figure 9-84. DMA Current Descriptor Address Register [0–3] (DMACDAR*x*)

Table 9-68 describes DMACDAR*x* fields.

Table 9-68. DMACDAR*x* Field Descriptions

Bits	Name	Description
31–5	CDA	Current descriptor address. Contains the current descriptor address of the segment descriptor in memory. It must be aligned on an 8-word boundary.
4	SNEN	Snoop enable. When set will allow snooping on DMA transactions.
3	EOSIE	End-of-segment interrupt enable. When set will generate an interrupt if the current DMA transfer for the current descriptor is finished.
2–0	—	Reserved, should be cleared

9.13.1.6.4 DMA Source Address Registers [0–3] (DMASAR*x*)

The source address register, shown in Figure 9-85, indicates the address where the DMA controller will be reading data from. This address can be in either PCI memory or 60x memory. The software has to ensure that this is a valid memory address.

The choice between PCI or 60x is done according to the following rule: If the address hits one of the PCI outbound windows, then the source data is read from the PCI memory. Otherwise, it is read from the 60x memory. Refer to Figure 9-13.

	31 16
Field	SA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10512(DMASAR0); 0x10592 (DMASAR1); 0x10612 (DMASAR2); 0x10692 (DMASAR3)

	15 0
Field	SA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10510 (DMASAR0); 0x10590 (DMASAR1); 0x10610 (DMASAR2); 0x10690 (DMASAR3)

Figure 9-85. DMA Source Address Register [0–3] (DMASAR*x*)

Table 9-69 describes DMASAR*x* fields.

Table 9-69. DMASAR*x* Field Descriptions

Bit	Name	Description
31–0	SA	Source address of DMA transfer. The content is updated after every DMA read operation.

9.13.1.6.5 DMA Destination Address Register [0–3] (DMADAR*x*)

The destination address register, shown in Figure 9-86, indicates the address where the DMA controller will be writing data to. This address can be in either PCI memory or 60x memory. The software has to ensure that this is a valid memory address.

The choice between PCI or 60x is done according to the following rule: If the address hits one of the PCI outbound windows, the destination data is written to the PCI memory. Otherwise, it is written to the 60x memory. Refer to Figure 9-13.

	31–16
Field	DA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x1051A (DMAADAR0); 0x1059A (DMAADAR1); 0x1061A (DMAADAR2); 0x1069A (DMAADAR3)

	15–0
Field	DA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10518 (DMAADAR0); 0x10598 (DMAADAR1); 0x10618 (DMAADAR2); 0x10698 (DMAADAR3)

Figure 9-86. DMA Destination Address Register [0–3] (DMADAR*x*)

Table 9-70 describes DMADAR*x* fields.

Table 9-70. DMADAR*x* Field Descriptions

Bit	Name	Description
31–0	DA	Destination address. The content is updated after every DMA write operation.

9.13.1.6.6 DMA Byte Count Register [0–3] (DMABCR*x*)

This register contains the number of bytes per transfer (maximum transfer size is 64 Mbytes).

	31–26	25–16
Field	—	BC
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10522 (DMABCR0); 0x105A2(DMABCR1); 0x10622 (DMABCR2); 0x106A2 (DMABCR3)	

	15–0
Field	BC
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10520 (DMABCR0); 0x105A0 (DMABCR1); 0x10620 (DMABCR2); 0x106A0 (DMABCR3)

Figure 9-87. DMA Byte Count Register [0–3] (DMABCR*x*)

Table 9-71 describes DMABCR*x* fields.

Table 9-71. DMABCR*x* Field Descriptions

Bit	Name	Description
31–26	—	Reserved, should be cleared
25–0	BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation.

9.13.1.6.7 DMA Next Descriptor Address Registers [0–3] (DMANDAR*x*)

The next descriptor address register (NDAR) 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 that the current descriptor register is pointing to. Refer to Figure 9-89.

	31–16
Field	NDA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10526 (DMANDAR0); 0x105A6 (DMANDAR1); 0x10626 (DMANDAR2); 0x106A6 (DMANDAR3)

	15–5	4	3	2–1	0
Field	NDA	NDSNEN	NDEOSIE	—	EOTD
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x10524 (DMANDAR0); 0x105A4 (DMANDAR1); 0x10624 (DMANDAR2); 0x106A4 (DMANDAR3)				

Figure 9-88. DMA Next Descriptor Address Registers [0–3] (DMANDAR*x*)

Table 9-72 describes DMANDAR*x* fields.

Table 9-72. DMANDAR*x* Field Descriptions

Bit	Name	Description
31–5	NDA	Next descriptor address. Contains the next descriptor address of the segment descriptor in memory. It must be aligned on an 8-word (32-byte) boundary.
4	NDSNEN	Next descriptor snoop enable. When set will allow snooping on DMA transactions.
3	NDEOSIE	Next descriptor end-of-segment interrupt enable. When set will generate an interrupt at the end of this DMA transfer.
2–1	—	Reserved, should be cleared
0	EOTD	End-of-transfer descriptor. When set indicates that this descriptor is the last one to be executed.

9.13.2 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 60x or PCI memory and are linked together into chains using the next-descriptor-address field.

Table 9-73. 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 source address register. For the bit definition, refer to Section 9.13.1.6.4, “DMA Source Address Registers [0–3] (DMASARx).”
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 destination address register. For the bit definition, refer to Section 9.13.1.6.5, “DMA Destination Address Register [0–3] (DMADARx).”
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 next descriptor address register. For the bit definition, refer to Section 9.13.1.6.7, “DMA Next Descriptor Address Registers [0–3] (DMANDARx).”
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 byte count register. For the bit definition, refer to Section 9.13.1.6.6, “DMA Byte Count Register [0–3] (DMABCRx).”

Application software initializes the current descriptor address register (DMACDAR*x*) 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).

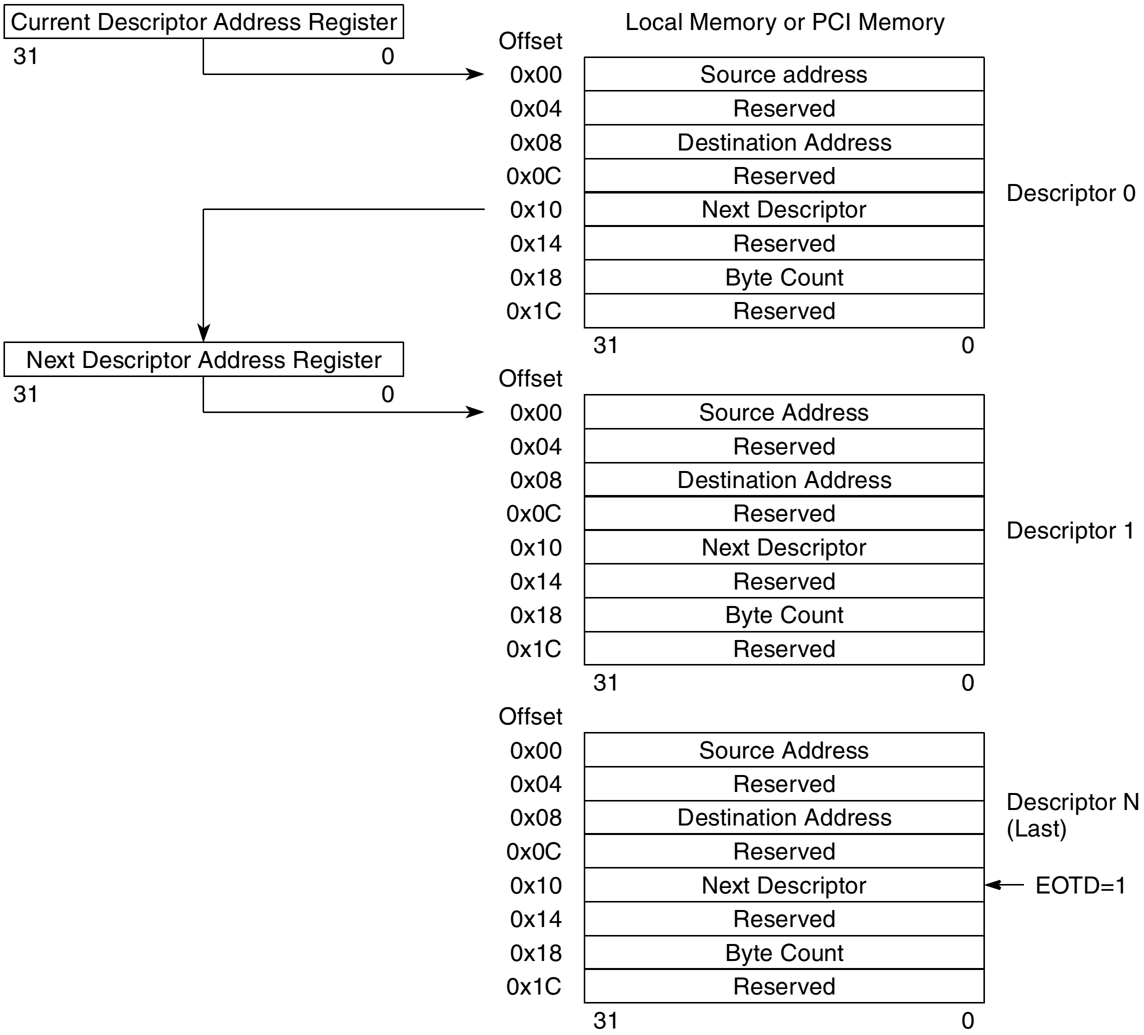


Figure 9-89. DMA Chain of Segment Descriptors

9.13.2.1 Descriptor in Big-Endian Mode

In big-endian mode, the descriptor in 60x memory should be programmed such that data appears in ascending significant-byte order. If segment descriptors are written to memory located in the 60x bus, 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>
```

9.13.2.2 Descriptor in Little-Endian Mode

In little-endian mode, the descriptor in PCI memory should be programmed such that data appears in descending significant byte order. If segment descriptors are written to memory located in the PCI bus, they are obeying the rules for little-endian mode.

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>
```

9.14 Error Handling

The PCI bridge provides error detection and reporting. This section describes how the PCI bridge handles different error (or interrupt) conditions.

Errors detected by the PCI bridge are reported by asserting internal error signals for each detected error. The system error ($\overline{\text{SERR}}$) and parity error ($\overline{\text{PERR}}$) signals are used to report errors on the PCI bus.

The PCI command and status registers and the error handling registers enable or disable the reporting and detection of specific errors. There are six registers that define capture and control functionality under error conditions. Refer to Section 9.11.1.9 through Section 9.11.1.14.

The PCI bridge detects illegal transfer sizes to its configuration registers, PCI master-abort cycles, PCI received target-abort errors, PCI parity errors, and overflow/underflow errors in the message unit.The PCI bridge latches the address and type of transaction that caused the error in the error registers to assist diagnostic and error handling software.

9.14.1 Interrupt and Error Signals

Although Section 9.11, "Configuration Registers," contains the definitions for the interrupt and error signals, this section describes the interactions between system components when an interrupt or error signal is asserted.

9.14.1.1 PCI Bus Error Signals

The PCI bridge uses two error signals to interact with the PCI bus, $\overline{\text{SERR}}$ and $\overline{\text{PERR}}$.

9.14.1.1.1 System Error ($\overline{\text{SERR}}$)

The $\overline{\text{SERR}}$ signal is used to report PCI address parity errors. It is driven for a single PCI clock cycle by the agent that is reporting the error. The agent responsible for driving AD[31:0] on a given PCI bus phase is responsible for driving even parity one PCI clock later on the PAR signal. (That is, the number of 1s on AD[31:0], PCI_C/$\overline{\text{BE}}$[3:0], and PAR equals an even number.) The target agent is not allowed to terminate with retry or disconnect if $\overline{\text{SERR}}$ is activated due to an address parity error.

Bits 8 and 6 of the PCI command register controls whether the PCI bridge asserts $\overline{\text{SERR}}$ upon detecting one of the error conditions.

9.14.1.1.2 Parity Error ($\overline{\text{PERR}}$)

The $\overline{\text{PERR}}$ signal is used to report PCI data parity errors during all PCI transactions, except for a PCI special-cycle command. The agent responsible for driving AD[31:0] on a given PCI bus phase is responsible for driving even parity one PCI clock later on the PAR signal. That is, the number of 1s on AD[31:0], PCI_C/$\overline{\text{BE}}$[3:0], and PAR equals an even number.

The $\overline{\text{PERR}}$ signal must be asserted by the agent receiving data two PCI clocks following the data phase for which a data parity error was detected. Only the master may report a read data parity error and only the selected target may report a write data parity error.

Bit 6 of the PCI command register controls whether the PCI bridge ignores $\overline{\text{PERR}}$.

9.14.1.1.3 Error Reporting

The error signals generated by the PCI bridge indicate which specific error has been detected.

The error control and address registers and the data capture registers are used to provide additional information about the detected error. When an error is detected, the associated information is latched inside these registers until all the associated error flags are cleared. Subsequent errors will set the appropriate error flags in the error detection registers, but will not latch additional information.

9.14.1.2 Illegal Register Access Error

An illegal register access error occurs when an access to a configuration register is not specified to be 1 beat. When this occurs, ESR[IRA] is set (refer to Section 9.11.1.9, “Error Status Register (ESR)”). If a read transaction causes the illegal access error the PCI bridge returns 0xFF (all 1s) and a write transaction with an illegal register access error will be dropped.

9.14.1.3 PCI Interface

The PCI bridge supports the error detection and reporting mechanism as specified in the *PCI Local Bus Specification, Revision 2.2*. The PCI bridge detects master and target abort errors, address parity errors, received $\overline{\text{SERR}}$, and master and target $\overline{\text{PERR}}$ errors. In these cases, the appropriate bit is set in the ESR, and the address, data and control information about the transaction is loaded in the PCI error address capture register (PCI_EACR), the PCI data capture register (PCI_EDCR) and the PCI error control capture register.

9.14.1.3.1 Address Parity Error

If the PCI bridge is acting as a PCI master and the target detects and reports (by asserting $\overline{\text{SERR}}$) a PCI address parity error, the PCI bridge sets bit 5 of the ESR and sets the detected parity error bit (bit 15) in the PCI status register. This setting of bit 15 is independent of the settings in the PCI command register.

If the PCI bridge is acting as a PCI target and detects a PCI address parity error, the PCI interface of the PCI bridge sets the status bit in the PCI status register (bit 15) and bit 0 of the ESR. If bits 6 and 8 of the PCI command register are set, the PCI bridge reports the address parity error by asserting $\overline{\text{SERR}}$ to the master (2 clocks after the address phase) and sets bit 14 of the PCI status register.

9.14.1.3.2 Data Parity Error

If the PCI bridge is acting as a PCI master and a write data parity error is signaled by the target asserting $\overline{\text{PERR}}$, the PCI bridge sets bit 8 of the PCI status register if the parity error response bit (bit 6) in the PCI command register is set. The PCI bridge sets bit 7 of the error status register (refer to Section 9.11.1.9, “Error Status Register (ESR)”), regardless of the configuration of the PCI command register.

If the PCI bridge is acting as a PCI master and a read data parity error occurs, the PCI bridge sets bit 8 of the PCI status register if the parity error response bit (bit 6) in the PCI command register is set. The PCI bridge sets bit 2 of the error status register. If the PCI command register of the PCI bridge is programmed to respond to parity errors (bit 6 of the PCI command register is set) the PCI bridge reports the error to the PCI target by asserting $\overline{\text{PERR}}$ and tries to complete the command if possible. The PCI bridge also sets bit 15 of the PCI status register regardless of the value of the parity error response bit (bit 6) in the PCI command register.

If the PCI bridge is acting as a PCI target when the write data parity error occurs, the PCI bridge sets bit 15 of the PCI status register and bit 1 of the error status register (ESR). The setting of these bits is independent of the settings in the PCI command register. If bit 6 of the PCI command Register is set, the PCI bridge asserts $\overline{\text{PERR}}$. When the data has all been transferred, the PCI bridge completes the operation but ignores the data.

If the PCI bridge is acting as a PCI target when the master asserts $\overline{\text{PERR}}$, the PCI bridge sets bit 6 of ESR (refer to Section 9.11.1.9, “Error Status Register (ESR)”), regardless of the configuration of the PCI command register.

9.14.1.3.3 Master-Abort Transaction Termination

If the PCI bridge, acting as a master, initiates a PCI bus transaction (excluding special-cycle transactions) but there is no response from any PCI agent ($\overline{\text{DEVSEL}}$ has not been asserted within five PCI bus clocks from the start of the address phase), the PCI bridge terminates the transaction with a master-abort and sets the master-abort flag (bit 13) in the PCI status register and bit 3 in the ESR.

In the case of no response for a PCI read configuration transaction, the PCI bridge terminates the transaction with a master-abort, but will return data of all ones and will not assert a machine check. This kind of termination enables the host CPU to perform a PCI device scan without having to know in advance if a particular PCI slot is populated or empty. The software still needs to mask the PCI_NO_RSP bit in the error mask register (refer to Section 9.11.1.10, “Error Mask Register (EMR)”). Any other type of transaction that is terminated with a master-abort results in a machine check interrupt.

9.14.1.3.4 Target-Abort Error

If a PCI transaction initiated by the PCI bridge is terminated by target-abort, the PCI bridge sets the received target-abort flag (bit 12) of the PCI status register and bit 4 of the error status register (refer to Section 9.11.1.9, “Error Status Register (ESR)”). Note that data transferred in a target-aborted transaction may be corrupt.

9.14.1.3.5 NMI

This signal is captured in bit 11 of the ESR (refer to Section 9.11.1.9, “Error Status Register (ESR)”). It indicates that an error has occurred on the 60x bus in a transaction that was originally initiated by the PCI bridge.

9.14.1.4 Embedded Utilities

Embedded utilities errors are errors detected in the I_2O interface. Embedded utilities errors are limited to queue overflows in the I_2O outbound free queue and the inbound post queue.

9.14.1.4.1 Outbound Free Queue Overflow

If the PCI bridge detects an I_2O outbound free queue overflow, it sets bit 8 of the error status register (refer to Section 9.11.1.9, “Error Status Register (ESR)”) and freezes all I_2O state information.

9.14.1.4.2 Inbound Post Queue Overflow

If the PCI bridge detects an I_2O inbound post queue overflow, it sets bit 9 of the error status register (refer to Section 9.11.1.9, “Error Status Register (ESR)”) and freezes all I_2O state information.

9.14.1.4.3 Inbound DoorBell Machine Check

If an external PCI master writes the inbound doorbell register such that the most significant bit is set, then bit 12 of ESR (refer to Section 9.11.1.9, “Error Status Register (ESR)”) is set and a machine check is asserted to the local processor.

Chapter 10
PLL and Clock Generator

The MPC8272’s clocking architecture includes two PLLs—the main PLL and the core PLL. The main PLL, together with the dividers, provides the internal 60x bus clock and internal clocks for all blocks in the chip except core blocks. The core PLL provides the internal core clocks.

The MPC8272’s clocking is a configurable system supporting two clock configuration modes. The clock configuration mode is set during power-on reset.

CLKIN is the primary timing reference for the MPC8272. The frequency of CLKIN equals the 60x bus frequency. The main PLL multiplies the frequency of the input clock to the final CPM frequency. Refer to Section 10.6, “Clock Configuration Modes.”

10.1 MPC8272 Clock Block Diagram

The MPC8272 clocking system, shown in Figure 10-1, is designed around two PLLs: the main PLL and the core PLL. The main PLL receives CLKIN as its input clock and multiplies it to provide MAIN_CLK, which is twice the CPM clock, to the clock block dividers. The dividers shown in Figure 10-1 generate all MPC8272 internal clocks by synchronously dividing MAIN_CLK. These clocks are then output from the clock block to the entire MPC8272.

10.1.1 Main PLL

The main PLL performs frequency multiplication and skew elimination. It allows the CPM to operate at a high internal clock frequency while using a low-frequency clock input. This has two immediate benefits:

- A lower clock input frequency reduces overall electromagnetic interference generated by the system.
- Oscillating at different frequencies eliminates the need for another oscillator.

10.1.2 Core PLL

The core PLL has the same advantages as the main PLL; it performs frequency multiplication and skew elimination for the core blocks. The core PLL input clock is synchronous with the 60x bus clock. Its configuration word, CORE_PLL_CFG[0–4], is determined by the MPC8272 clock configuration mode setting. According to the setting, the core PLL multiplies the internal bus clock and synchronously provides the core clocks.

10.1.3 Skew Elimination

The PLL can tighten synchronous timings by eliminating skew between phases of the internal clock and the external clock entering the chip (CLKIN). Skew elimination is always active when the PLL is enabled. Disabling the PLL (PLL bypass) can greatly increase clock skew.

10.1.4 Dividers

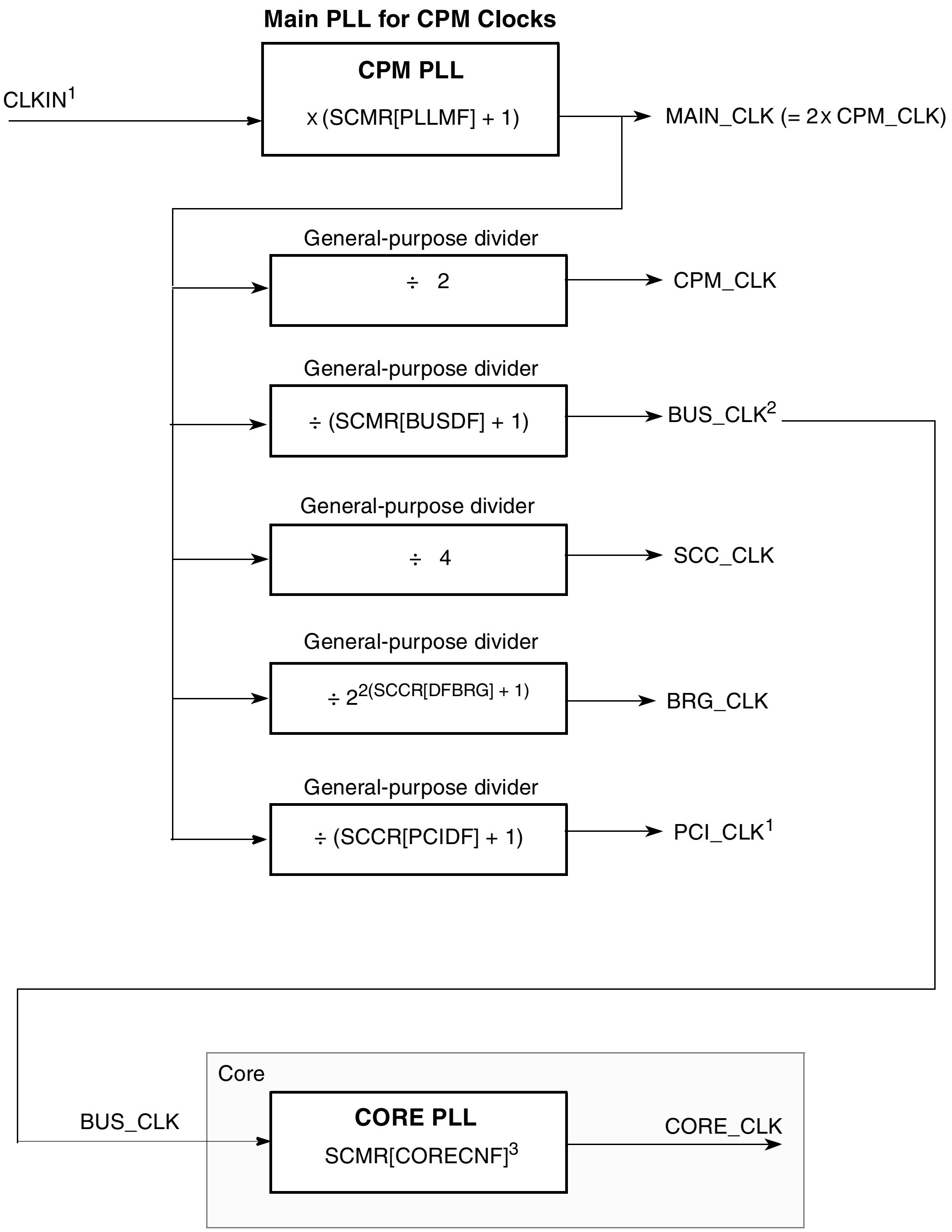
The PLL output clock, MAIN_CLK, is twice the CPM clock. MAIN_CLK applies to general-purpose dividers. Each MPC8272 internal clock is generated by a dedicated divisor that is a programmable number between 1 and 16. Dividers are determined by the clock configuration modes that are selected by seven bits during power-on reset—three hardware configuration pins (MODCK[1–3]) and four bits from the hardware configuration word[28–31] (MODCK_H). For complete lists of these dividers, refer to the *MPC8272 Hardware Specifications* (order number: MPC8272EC). Note that all dividers’ output clocks have identical skew in relation to the input clock because the delay through the dividers for all clocks is identical independent of how its dividers have been programmed.

10.1.5 Internal Clock Signals

The internal logic of the MPC8272 generates the next internal clock lines:

- CPM general system clocks (CPM_CLK)
- 60x bus (BUS_CLK). Identical to CLKIN
- SCC clocks (SCC_CLK)
- Baud-rate generator clock (BRG_CLK)
- PCI clock (PCI_CLK)
- DLL clocks

The PLL synchronizes these clock signals to each other.



Notes:

[1] In PCI agent mode, CLKIN = PCI_CLK. Refer to Section 10.1.6, "PCI Bridge as an Agent Operating from the PCI System Clock."

[2] BUS_CLK = CLKIN.

[3] The core PLL multiplication is set through SCMR[CORECNF] as described in Table 10-3.

[4] SCMR is a read-only register. Its value is determined during power-on reset ($\overline{\text{PORESET}}$). Refer to Section 10.5, "System Clock Mode Register (SCMR)."

Figure 10-1. MPC8272 System Clock Architecture

10.1.6 PCI Bridge as an Agent Operating from the PCI System Clock

If the MPC8272 is connected to a system that generates the PCI clock, the PCI clock should be fed to the CLKIN1 pin. The PCI clock is internally multiplied by the PLL to generate the chip's internal high-speed clock. This clock is used to generate the 60x bus clock. The 60x bus clock is then driven by a DLL circuit to the DLLOUT pin, which has a feedback path from the board to the CLKIN2 pin. This feedback clock signal is used by the DLL logic to minimize clock skew between the internal and external clocks.

NOTE

All PCI timings are measured relative to CLKIN1; all 60x bus timings are measured relative to CLKIN2.

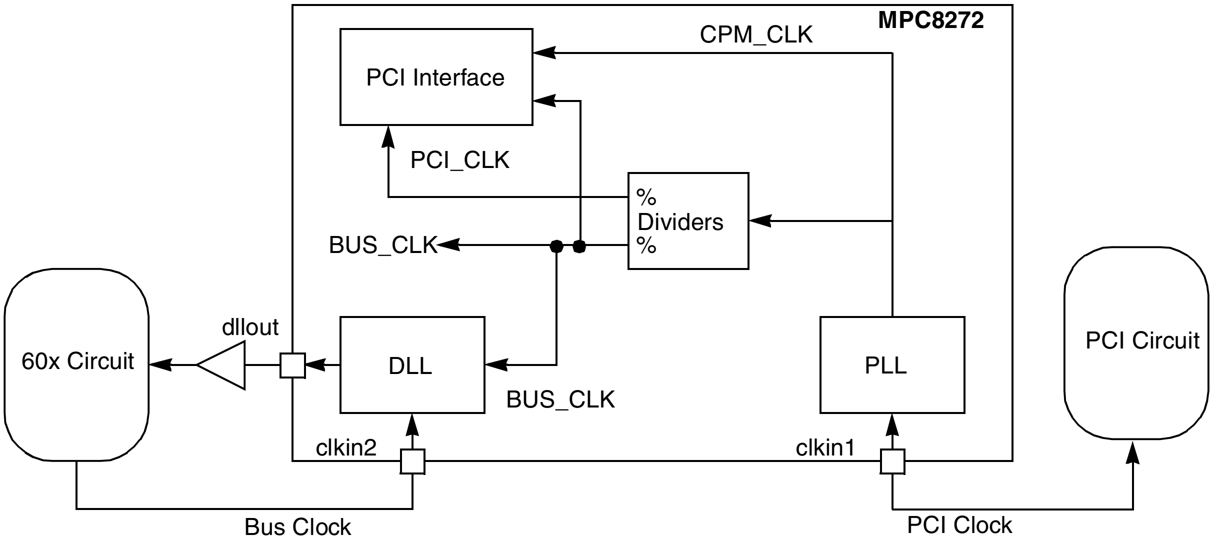


Figure 10-2. PCI Bridge as an Agent Operating from the PCI System Clock

10.1.7 PCI Bridge as a Host Generating the PCI System Clock

In a system where the MPC8272 is the host that generates the PCI clock, the 60x bus clock should be driven to the CLKIN1 pin. The 60x bus clock is internally multiplied by the PLL to generate the CPM high-speed clock and then internally divided to generate the PCI bus clock. Next the PCI bus clock is driven by the DLL circuit to the DLLOUT pin, which has a feedback path from the board to the CLKIN2 pin. This feedback controls clock skew by ensuring the same internal and external clock timing.

NOTE

All PCI timings are measured relative to CLKIN2, and all 60x bus timings are measured relative to CLKIN1.

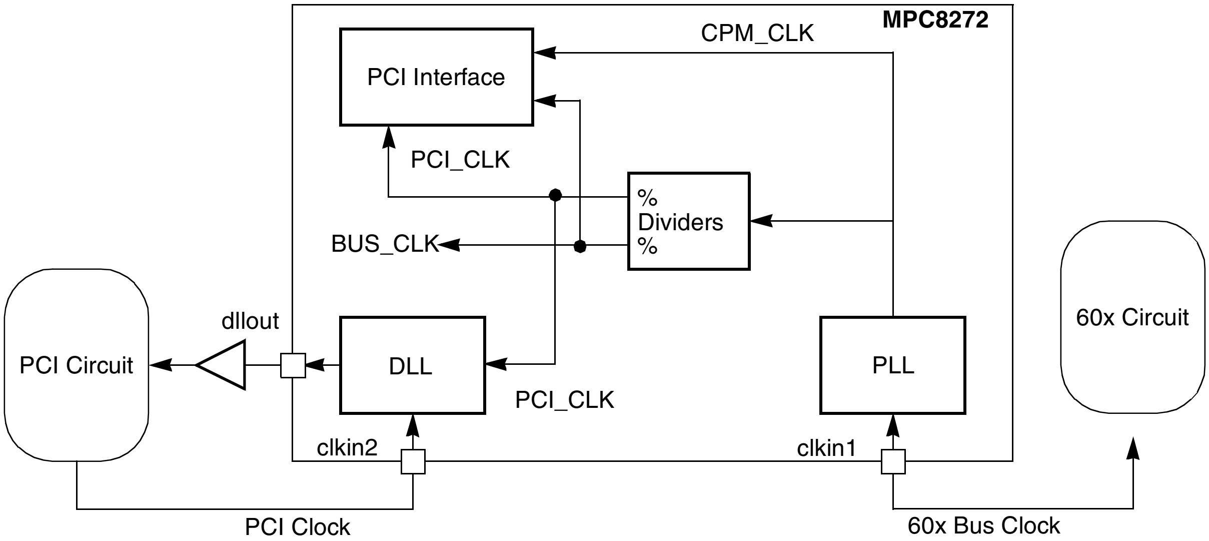


Figure 10-3. PCI Bridge as a Host, Generating the PCI System Clock

10.2 External Clock Inputs

The input clock source to the PLL is an external clock oscillator at the bus frequency. The PLL skew elimination between the CLKIN pin and the internal bus clock is guaranteed.

10.3 PLL Pins

Table 10-1 shows the dedicated PLL pins.

Table 10-1. Dedicated PLL Pins

Signal	Description
VCCSYN1	Drain voltage—Analog VDD dedicated to core analog PLL circuits. To ensure core clock stability, filter the power to the VCCSYN1 input with a circuit similar to the one in Figure 10-4. To filter as much noise as possible, place the circuit as close as possible to VCCSYN1. The 0.1-µF capacitor should be closest to VCCSYN1, followed by the 10-µF capacitor, and finally the 10-Ω resistor to Vdd. These traces should be kept short and direct.
VCCSYN	Drain voltage—Analog VDD dedicated to analog main PLL circuits. To ensure internal clock stability, filter the power to the VCCSYN input with a circuit similar to the one in Figure 10-4. To filter as much noise as possible, place the circuit should as close as possible to VCCSYN. The 0.1-µF capacitor should be closest to VCCSYN, followed by the 10-µF capacitor, and finally the 10-Ω resistor to Vdd. These traces should be kept short and direct.

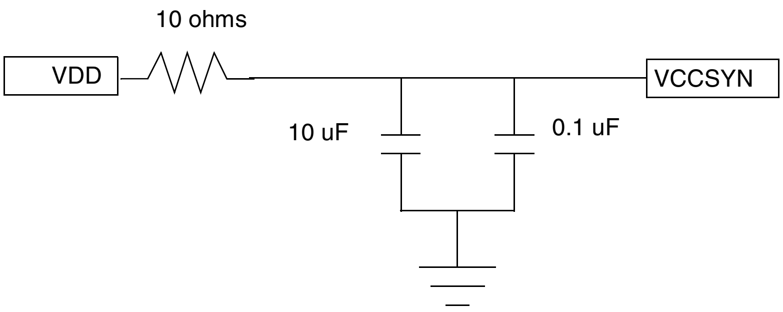


Figure 10-4. PLL Filtering Circuit

10.4 System Clock Control Register (SCCR)

The system clock control register (SCCR), shown in Figure 10-5, is memory-mapped into the MPC8272’s internal space.

	0 22	23	24	25 28	29	30 31
Field	—	—	PCI_MODCK	PCIDF	CLPD	DFBRG
Reset	0	1	Refer to Table 10-2.		0	01
R/W	R/W			R	R/W	
Addr	0x10C80					

Figure 10-5. System Clock Control Register (SCCR)

Table 10-2 describes SCCR fields.

Table 10-2. SCCR Field Descriptions

Bits	Name	Defaults		Description
		$\overline{\text{PORESET}}$	Hard Reset	
0–22	—	0	Unaffected	Reserved
23	—	1	Unaffected	Reserved
24	PCI_MODCK	PCI_MODCK	Unaffected	Reflects the value of the PCI_MODCK bit from the hard reset configuration word
25–28	PCIDF	Config pins[1]	Unaffected	PCI division factor

Table 10-2. SCCR Field Descriptions (continued)

Bits	Name	Defaults		Description
		PORESET	Hard Reset	
29	CLPD	0	Unaffected	CPM low power disable. 0 Default. CPM does not enter low power mode when the core enters low power mode. 1 CPM and SIU enter low power mode when the core does. This may be useful for debug tools that use the assertion of QREQ as an indication of breakpoint in the core.
30–31	DFBRG	01	Unaffected	Division factor of BRG_CLK. Determines BRG_CLK frequency. Changing the value does not result in a loss of lock condition. BRG_CLK is divided from the PLL output clock (defined as “MAIN_CLK”), which is 2x the CPM clock. Refer to Figure 10-1. The decimal equivalent of the binary value of SCCR[30–31] determines the overall BRG_CLK dividers, as shown in Figure 10-1. MAIN-CLK is divided by $2^{2(\text{DFBRG} + 1)}$. 00 Decimal value of 0; MAIN_CLK divided by 4 01 Decimal value of 1; MAIN_CLK divided by 16 (normal operation) 10 Decimal value of 2; MAIN_CLK divided by 64 11 Decimal value of 3; MAIN_CLK divided by 256

[1] MODCK[1-3] and MODCK_H. Refer to Section 10.1.4, “Dividers.”

10.5 System Clock Mode Register (SCMR)

The PLL, low power, and reset control register (SCMR), shown in Figure 10-6, hold the parameters necessary for determining the output clock frequencies. To understand the interaction of these values, refer to Section 10.1, “MPC8272 Clock Block Diagram.”

	0	2	3 7	8 11	12 15
Field	PLLBP	—	CORECNF	BUSDF	CPMDF
Reset	—	00	Refer to Table 10-3		
R/W	R				
Addr	0x10C88				

	16 27	28 31
Field	—	PLLMF
Reset	0000_0000_0000	Refer to Table 10-3.
R/W	R	
Addr	0x10C8A	

Figure 10-6. System Clock Mode Register (SCMR)

Table 10-3 describes SCMR fields.

Table 10-3. SCMR Field Descriptions

Bits	Name	Defaults		Description
		$\overline{\text{PORESET}}$	Hard Reset	
0	PLLBP	Config pin[1]	Unaffected	Reset configuration for PLL bypass. 0 Normal operation 1 Bypass CPM PLL
1–2	—	—	—	Reserved
3–7	CORECNF	Config pins	Unaffected	Core PLL configuration. Refer to Table 10-4 to see how these bits translate to the actual core PLL multiplication mode.
8–11	BUSDF	Config pins	Unaffected	60x bus division factor
12–15	CPMDF	Config pins	Unaffected	CPM division factor. This value is always 1.
16–27	—	—	—	Reserved
28–31	PLLMF	Config pins	Unaffected	PLL multiplication factor. PLLMF controls the value of the divider in the PLL feedback loop. **Note:** The definition of PLLMF depends on the clock mode: • PCI host mode: PLLMF = 2(CPM_CLK/**CLKIN**)[2] – 1 • PCI agent mode: PLLMF = 2(CPM_CLK/**PCI_CLK**)[3] – 1 Refer to Sections 10.1.6 and 10.1.7 for more details.

[1] MODCK[1-3] and MODCK_H. Refer to Section 10.1.4, “Dividers.”

[2] (CPM_CLK/CLKIN) is identified as the CPM Multiplication Factor in the section, “Clock Configuration Modes,” in the *MPC8272 Family Hardware Specifications*.

[3] (CPM_CLK/PCI_CLK) is identified as the CPM Multiplication Factor in the section, “Clock Configuration Modes,” in the *MPC8272 Family Hardware Specifications*.

10.5.1 SCMR[CORECNF] Definitions

Table 10-4 shows SCMR[CORECNF] bit values and translations to the core PLL mode.

Table 10-4. 60x Bus-to-Core Frequency

SCMR[CORECNF]	Bus-to-Core Multiplier
0x04, 0x05, 0x15	2
0x06, 0x11	2.5
0x08, 0x10	3
0x0E, 0x1E	3.5
0x0A, 0x1A	4
0x07, 0x17	4.5x
0x0B, 0x1B	5
0x09, 0x19	5.5x
0x0D, 0x1D	6
0x12	6.5x
0x14	7x
0x16	7.5x
0x1C	8x
0x03, 0x13	PLL off/bypassed
0x0F, 0x1F	PLL off

10.6 Clock Configuration Modes

The MPC8272 has two clocking modes: PCI host and PCI agent. The clocking mode is set according to two sources:

- $\overline{\text{PCI\_HOST\_EN}}$— An input signal. Refer to Chapter 6, “External Signals,” and Chapter 9, “PCI Bridge.”
- PCI_MODCK—Bit 27 in the hard reset configuration word. Refer to Section 5.4.1, “Hard Reset Configuration Word.”

In each clocking mode, the configuration of bus, core, PCI, and CPM frequencies is determined by seven bits during the power-on reset—three hardware configuration pins (MODCK[1–3]) and four bits from hardware configuration word[28–31] (MODCK_H). Both the PLLs and the dividers are set according to the selected MPC8272 clock operation mode. For further information and complete lists of each clock mode’s possible clock configurations, see Section 1.3, “Clock Configuration Modes,” in the *MPC8272 Family Hardware Specifications* (order number: MPC8272EC).

Chapter 11
Memory Controller

The memory controller is responsible for controlling a maximum of eight memory banks sharing a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and three user-programmable machines (UPMs). It supports a glueless interface to synchronous DRAM (SDRAM), SRAM, EEPROM, Flash EEPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. This flexible memory controller allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to synchronous DRAMs, using SDRAM pipelining, bank interleaving, and back-to-back page mode to achieve the highest performance.
- The GPCM provides interfacing for simpler, lower-performance memory resources and memory-mapped devices. The GPCM 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 address multiplexing of the external bus, refresh timers, and generation of programmable control signals for row address and column address strobes to allow for a glueless interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral. The refresh timers allow refresh cycles to be initiated. The UPM can be used to generate different timing patterns for the 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 request. Refresh timers are also available to periodically generate user-defined refresh cycles.

This chapter describes the 60x bus memory controller. The memory controller is identical to any other MPC8272 memory controller on the 60x bus.

The MPC8272 supports the following new features as compared to the MPC860 and MPC850.

- The synchronous DRAM machine enables back-to-back memory read or write operations using page mode, pipelined operation and bank interleaving for high-performance systems.
- The memory controller has eight memory banks as well as two SDRAM machines, three user-programmable machines (UPMs), and a GPCM.
- The memory controller supports atomic operation.
- One data buffer control
- Flexible UPM assignment—The user can assign any of the three UPMs to the 60x bus.

Figure 11-1 shows the dual-bus architecture.

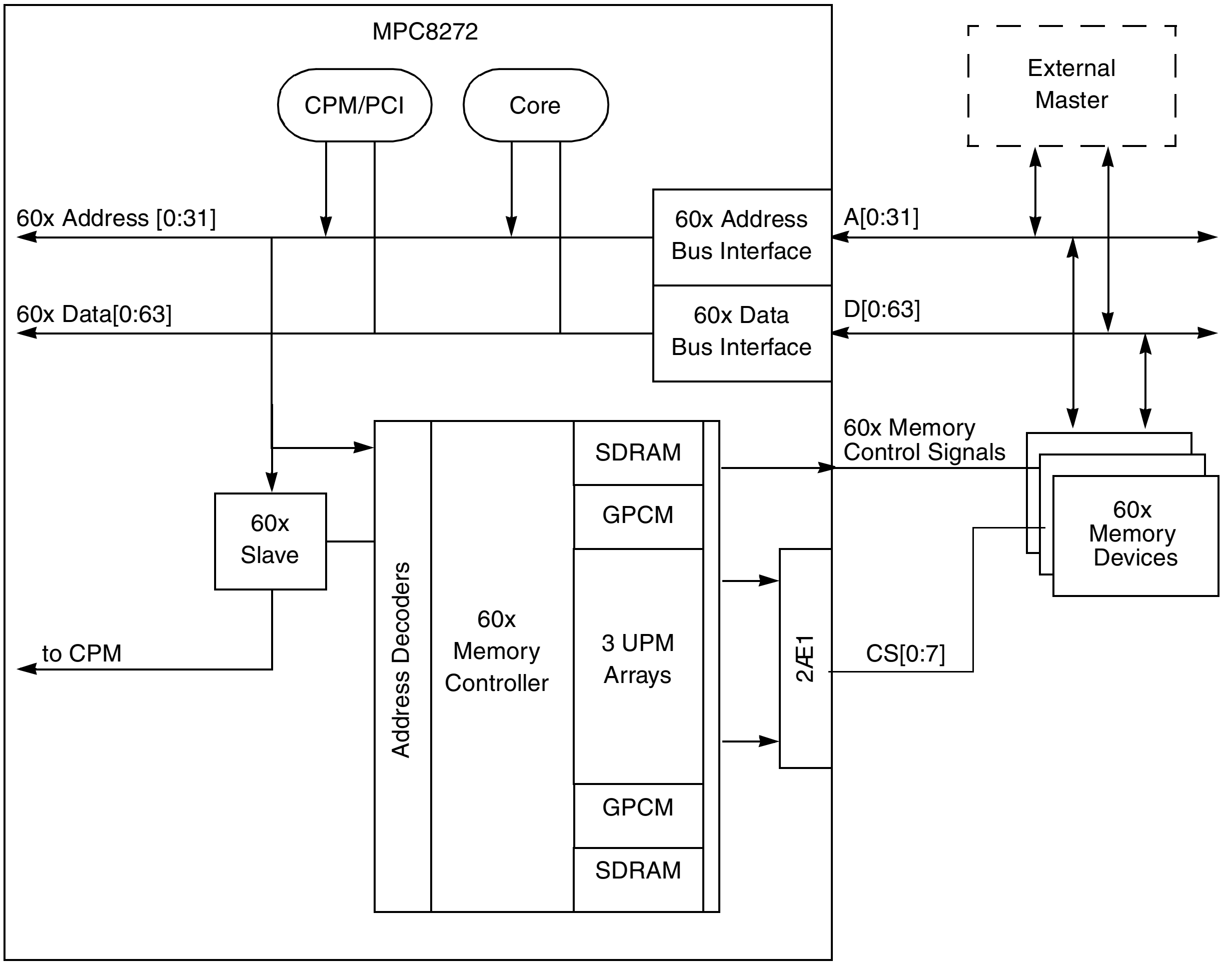


Figure 11-1. Memory Controller Architecture

11.1 Features

The memory controller’s main features are as follows:

- Eight memory banks
  - 32-bit address decoding with mask
  - Variable block sizes (32 Kbytes to 4 Gbytes)
  - Write-protection capability
  - Control signal generation machine selection on a per-bank basis
  - Supports internal or external masters on the 60x bus
  - Data buffer controls activated on a per-bank basis
  - Atomic operation
  - Extensive external memory-controller/bus-slave support
  - Data pipeline to reduce data setup time for synchronous devices

- Synchronous DRAM machine
  - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
  - Back-to-back page mode for consecutive, back-to-back accesses
  - Supports 2-, 4- and 8-way bank interleaving
  - Supports SDRAM port size of 64-bit, 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-, 32-, and 64-bit devices
  - Minimum two clock accesses to external device
  - Eight byte write enable signals ($\overline{\text{WE}}$)
  - Output enable signal ($\overline{\text{OE}}$)
  - External access termination signal ($\overline{\text{GTA}}$)
- Three UPMs
  - Each machine can be assigned to the 60x or memory banks.
  - 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 or external 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
    - Chip-select line
    - Byte-select lines
    - Six external general-purpose lines
  - Supports 8-, 16-, 32-, and 64-bit memory port sizes
  - Page mode support for successive transfers within a burst
  - Internal address multiplexing for all on-chip bus masters supporting 64-, 128-, 256-, and 512-Kbyte, and 1-, 2-, 4-, 8-, 16-, 32-, 64-, 128-, 256-Mbyte page banks

11.2 Basic Architecture

The memory controller consists of three basic machines:

- Synchronous DRAM machine
- General-purpose chip-select machine (GPCM)
- Three UPMs

Each bank can be assigned to any one of these machines through BRx[MS], as shown in Figure 11-2. The MS and MxMR[BSEL] bits (for UPMs) assign banks to the 60x bus as shown in Figure 11-2. Addresses are decoded by comparing (A[0:16] bit-wise and ORx[AM]) with BRx[BA]. If an address match occurs in multiple banks, the lowest numbered bank has priority.

When a memory address matches BRx[BA], the corresponding machine takes ownership of the external signals that control access and maintains control until the cycle ends.

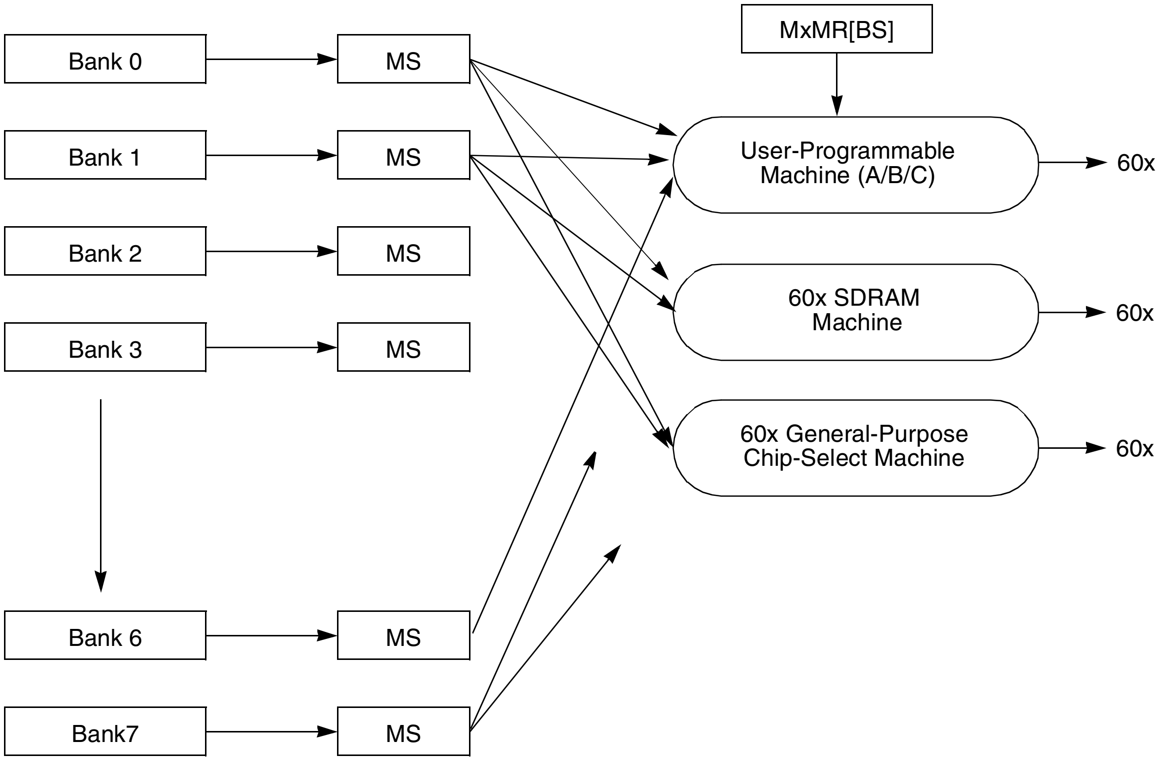


Figure 11-2. Memory Controller Machine Selection

Some features are common to all machines.

- A 17-bit most-significant address decode on each memory bank
- The block size of each memory bank can vary between 32 Kbytes (1 Mbyte for SDRAM) and 4 Gbytes (128 Mbytes for SDRAM).
- Each memory bank can be selected for read-only or read/write operation.
- Each memory bank can use data pipelining, which reduces the required data setup time for synchronous devices.
- Each memory bank can be controlled by an external memory controller or bus slave.

The memory controller functionality minimizes the need for glue logic in MPC8272-based systems. In Figure 11-3, $\overline{\text{CS0}}$ is used with the 16-bit boot EPROM with BR0[MS] defaulting to select the GPCM. $\overline{\text{CS1}}$ is used as the $\overline{\text{RAS}}$ signal for 64-bit DRAM with BR1[MS] configured to select UPMA. $\overline{\text{BS}}$[0:7] are used as $\overline{\text{CAS}}$ signals on the DRAM.

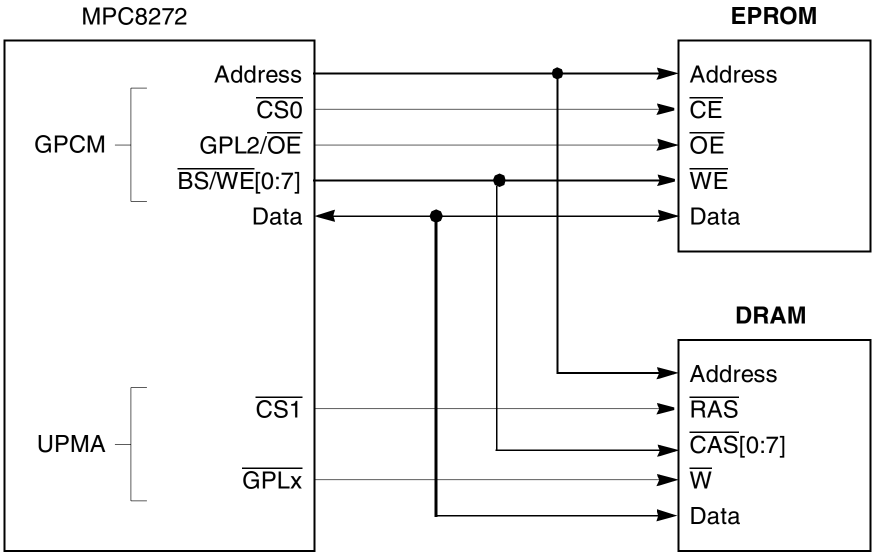


Figure 11-3. Simple System Configuration

Implementation differences between the supported machines are described below:

- The SDRAM machine provides a glueless interface to JEDEC-compliant SDRAM devices, and using SDRAM pipelining, page mode, and bank interleaving delivers very high performance. To allow fine tuning of system performance, the SDRAM machine provides two types of page modes selectable per memory bank:
  - — Page mode for consecutive back-to-back accesses (normal operation)
  - — Page mode for intermittent accesses

  SDRAM machines are available on the 60x; each memory bank can be assigned to any SDRAM machine.
- The GPCM provides a glueless interface to EEPROM, SRAM, flash EEPROM (FEPROM), and other peripherals. The GPCM is available on CS[0:11]. CS0 also functions as the global (boot) chip-select for accessing the boot EEPROM or FLASH device. The chip-select allows 0 to 30 wait states.
- The UPMs provide a flexible interface to many types of memory devices. Each UPM can control the address multiplexing for accessing DRAM devices and the timings of BS[0:7] and GPL. Each UPM can be configured to control memory devices on the 60x. Each memory bank can be assigned to any UPM.

  Each UPM is a programmable RAM-based machine. The UPM toggles the memory controller external signals as programmed in RAM when an internal or external master initiates any external read or write access. The UPM also controls address multiplexing, address increment, and transfer acknowledge (TA) assertion for each memory access. The UPM specifies a set of signal patterns for a user-specified number of clock cycles. The UPM RAM pattern run by the memory controller is selected according to the type of external access transacted. At every clock cycle, the logical value of the external signals specified in the RAM array is output on the corresponding UPM pins.

Figure 11-4 shows a basic configuration.

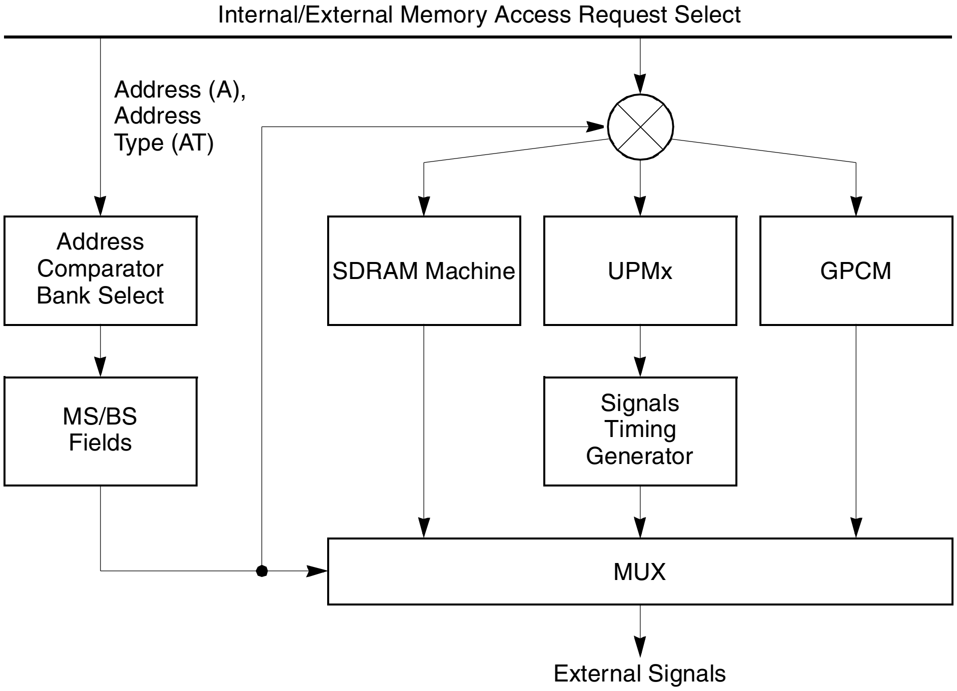

Internal/External Memory Access Request Select
Address (A), Address Type (AT)
Address Comparator Bank Select
MS/BS Fields
SDRAM Machine
UPMx
GPCM
Signals Timing Generator
MUX
External Signals

Figure 11-4. Basic Memory Controller Operation

The SDRAM mode registers (LSDMR and PSDMR) define the global parameters for the 60x SDRAM devices. Machine A/B/C mode registers (MxMR) define most of the global features for each UPM. GPCM parameters are defined in the option register (ORx). Some SDRAM and UPM parameters are also defined in ORx.

11.2.1 Address and Address Space Checking

The defined base address is written to the BRx. The bank size is written to the ORx. Each time a bus cycle access is requested on the 60x bus, addresses are compared with each bank. If a match is found on a memory controller bank, the attributes defined in the BRx and ORx 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).

11.2.2 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 ORx 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 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 ORx[PMS] is set.

11.2.3 Transfer Error Acknowledge ($\overline{\text{TEA}}$) Generation

The memory controller asserts the transfer error acknowledge signal ($\overline{\text{TEA}}$) in the following cases:

- An unaligned or burst access is attempted to internal MPC8272 space (registers or dual-port RAM). Note that the dual-port RAM cannot be accessed through bursts.
- A bus monitor timeout

11.2.4 Data Buffer Controls ($\overline{\text{BCTL}\textit{x}}$)

The memory controller provides two data buffer controls for the 60x bus ($\overline{\text{BCTL0}}$ and $\overline{\text{BCTL1}}$). These controls are activated when a GPCM- or UPM-controlled bank is accessed and can be disabled by setting ORx[BCTLD]. An access to an SDRAM-machine controlled bank does not activate the $\overline{\text{BCTLx}}$ controls. The $\overline{\text{BCTL}}$ signals are asserted on the rising edge of CLKIN on the first cycle of the memory controller operation. They are negated on the rising edge of CLKIN after the last assertion of $\overline{\text{PSDVAL}}$ of the access. (See Section 11.2.10, "Partial Data Valid Indication (PSDVAL).") If back-to-back memory controller operations are pending, $\overline{\text{BCTLx}}$ is not negated.

The BCTL signals have a programmable polarity. See Section 4.3.2.4, "SIU Module Configuration Register (SIUMCR)."

11.2.5 Atomic Bus Operation

The MPC8272 supports the following kinds of atomic bus operations BRx[ATOM]:

- Read-after-write (RAWA). When a write access hits a memory bank in which ATOM = 01, the MPC8272 locks the bus for the exclusive use of the accessing master (internal or external).

  While the bus is locked, no other device can be granted the bus. The lock is released when the master that created the lock accesses the same bank with a read transaction. If the master fails to release the lock within 256 bus clock cycles, the lock is released and a special interrupt is generated. This feature is intended for CAM operations.
- Write-after-read (WARA). When a read access hits a memory bank in which ATOM = 10, the MPC8272 locks the bus for the exclusive use of the accessing master (internal or external).

  During the lock period, no other device can be granted bus mastership. The lock is released when the device that created the lock access the same bank with a write transaction. If the device fails to release the lock within 256 bus clock cycles, the lock is released and a special interrupt is generated.

NOTE

This mechanism does not replace the PowerPC reservation mechanism.

11.2.6 Data Pipelining

Multiple-MPC8272 systems that utilize the 60x bus on its high frequency mode may face a timing problem when synchronous memories, such as SDRAM, are used. Because these devices can output data every cycle and because the data forwarding requires additional data setup time, the timing constraints are extremely hard to meet. In such systems, the user should set the data pipelining bit, BRx[DR]. This creates data pipelining of one stage within the memory controller in which the data forwarding is done, thus eliminating the additional data setup time requirement.

Note that this feature cannot be used with L2 cacheable banks and that in systems that involve both MPC8272-type masters and 60x compatible master, this feature can still be used on the 60x bus under the following restrictions:

1. The arbiter and the memory controller are in the same MPC8272.
2. The register field BCR[NPQM] is set up correctly.

See "Section 11.9, "External Master Support (60x-Compatible Mode)," and "Section 4.3.2.1, "Bus Configuration Register (BCR)."

11.2.7 External Memory Controller Support

The MPC8272 has an option to allocate specific banks (address spaces) to be controlled by an external memory controller or bus slave, while retaining all the bank properties: port size, data check/correction, atomic operation, and data pipelining. This is done by programming BRx and ORx[AM] and setting the external memory controller bit, BRx[EMEMC]. This action automatically assigns the bank to the 60x bus. For an access that hits the bank, all bus acknowledgment signals (such as $\overline{\mathrm{AACK}}$, $\overline{\mathrm{PSDVAL}}$, and $\overline{\mathrm{TA}}$) and the memory-device control strobes are driven by an external memory controller or slave. If the device that initiates the transaction is internal to the MPC8272, the memory controller handles the port size, data checking, atomic locking, and data pipelining as if the access were governed by it.

This feature allows multiple MPC8272 systems to be connected in 60x-compatible mode without losing functionality and performance. It also makes it easy to connect other 60x-compatible slaves on the 60x bus.

11.2.8 Data Buffer Control Signals ($\overline{\mathrm{BCTL1}}$ and $\overline{\mathrm{BCLT2}}$)

The memory controller provides two data buffer controls for the 60x bus. These controls can be activated on a per-bank basis by setting BRx[BCTL]. The BCTL signals are asserted on the rising edge of CLKIN on the first cycle of memory controller operation. They are negated on the rising edge of CLKIN after the last $\overline{\mathrm{TA}}$ of the access is asserted. BCTL signals are not negated if a back-to-back memory controller operation is pending.

BCTL signals have a programmable polarity, which is described in Section 4.3.2.4, "SIU Module Configuration Register (SIUMCR)."

11.2.9 External Address Latch Enable Signal (ALE)

The memory controller provides control for an external address latch, needed on the 60x bus in 60x compatible mode. ALE is asserted for one clock cycle on the first cycle of each memory-controller transaction. In this section, whenever ALE is not on a timing diagram, assume that it is asserted on the first cycle in which $\overline{\mathrm{CS}}$ can be asserted.

NOTE

ALE is relevant only in 60x-compatible mode.

11.2.10 Partial Data Valid Indication (PSDVAL)

The 60x bus has an internal 64-bit data bus. According to the 60x bus specification, TA is asserted when up to a double word of data is transferred. Because the MPC8272 supports memories with port sizes smaller than 64 bits, there is a need for partial data valid indication. The memory controller uses PSDVAL to indicate that data is latched by the memory on write accesses or valid data is present on read accesses. The quantity of the data depends on the memory port size and the transfer size. The memory controller accumulates PSDVAL assertions, and when a double word (or the transfer size) is transferred, the memory controller asserts TA to indicate that a 60x data beat was transferred. Table 11-1 shows the number of PSDVAL assertions needed for one TA assertion under various circumstances.

Table 11-1. Number of PSDVAL Assertions Needed for TA Assertion

Port Size	Transfer Size	PSDVAL Assertions	TA Assertions
64	Any	1	1
32	Double word	2	1
32	Word/half word/byte (32-bit aligned)	1	1
16	Double word	4	1
16	Word	2	1
16	Half/byte	1	1
8	Double word	8	1
8	Word	4	1
8	Half	2	1
8	Byte	1	1

Figure 11-5 shows a double-word transfer on 32-bit port size memory.

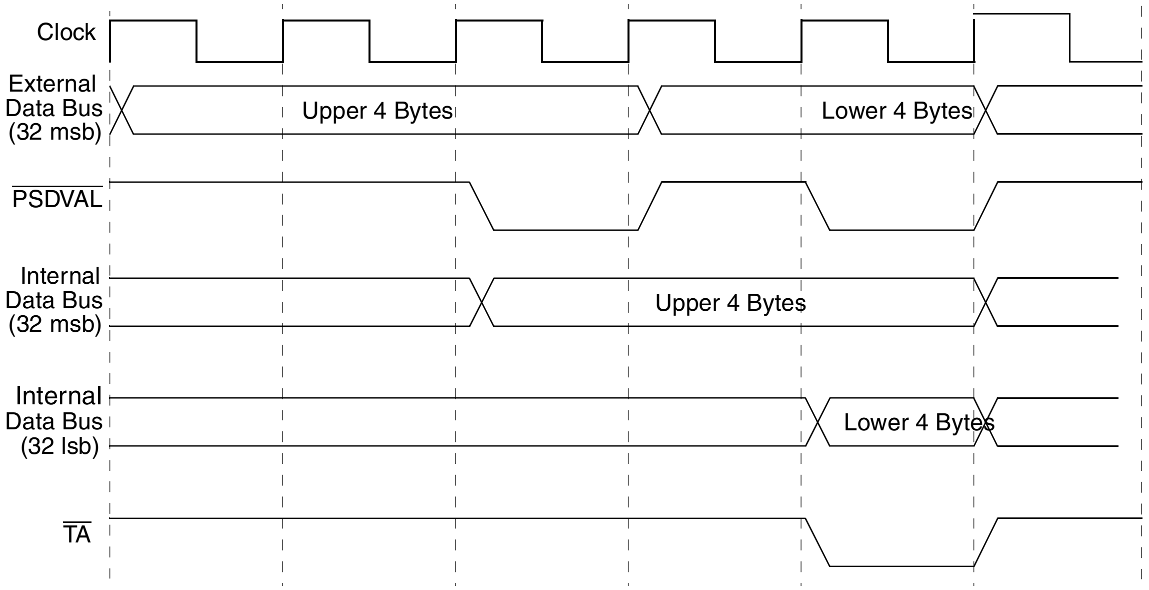

Clock
External Data Bus (32 msb)
Upper 4 Bytes
Lower 4 Bytes
PSDVAL
Internal Data Bus (32 msb)
Upper 4 Bytes
Internal Data Bus (32 lsb)
Lower 4 Bytes
TA

Figure 11-5. Partial Data Valid for 32-Bit Port Size Memory, Double-Word Transfer

11.2.11 BADDR[27:31] Signal Connections

The memory controller uses BADDR[27:31] to interface memory and peripheral devices on the 60x bus in 60x-compatible mode. Not all the BADDR line are necessarily used.

Use Table 11-2 to determine which BADDR lines are needed for the device connection.

Table 11-2. BADDR Connections

BADDR[x]	64/72-Bit Port Size SDRAM	Non-SDRAM 64-/72-Bit Port Size Device	32-Bit Port Size SDRAM	Non-SDRAM 32-Bit Port Size Device	Any 16-Bit Port Size Device	Any 8-Bit Port Size Device
BADDR[27]	N.C.	Connected	N.C.	Connected	Connected	Connected
BADDR[28]	N.C.	Connected	N.C.	Connected	Connected	Connected
BADDR[29]	N.C.	N.C	N.C.	Connected	Connected	Connected
BADDR[30]	N.C.	N.C	N.C.	N.C	Connected	Connected
BADDR[31]	N.C.	N.C	N.C.	N.C.	N.C.	Connected

11.3 Register Descriptions

Table 11-3 lists registers used to control the 60x bus memory controller.

Table 11-3. 60x Bus Memory Controller Registers

Abbreviation	Name	Reference (Section/Page)
BR0–BR7	Base register banks 0–7	11.3.1/11-11
OR0–OR7	Option register banks 0–7	11.3.2/11-13
PSDMR	60x bus SDRAM machine mode register	11.3.3/11-18
MAMR	UPMA mode register	11.3.4/11-21
MBMR	UPMB mode register	
MCMR	UPMC mode register	
MDR	Memory data register	11.3.5/11-23
MAR	Memory address register	11.3.6/11-24
MPTPR	Memory refresh timer prescaler register	11.3.9/11-26
PURT	60x bus assigned UPM refresh timer	11.3.7/11-25
PSRT	60x bus assigned SDRAM refresh timer	11.3.8/11-25
TESCRx	60x bus error status and control registers	11.3.10/11-26

11.3.1 Base Registers (BR*x*)

The base registers (BR0–BR7) contain the base address and address types that the memory controller uses to compare the address bus value with the current address accessed. Each register also includes a memory attribute and selects the machine for memory operation handling. Figure 11-7 shows the BRx register format.

	0 15
Field	BA
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10100 (BR0); 0x10108 (BR1); 0x10110 (BR2); 0x10118 (BR3); 0x10120 (BR4); 0x10128 (BR5); 0x10130 (BR6); 0x10138 (BR7)

	16	17 18	19 20	21 22	23	24 26	27	28 29	30	31
Field	BA	—	PS	—	WP	MS	EMEMC	ATOM	DR	V [1]
Reset	000		See note	0000_000			See note	000		See note
R/W	R/W									
Addr	0x10102 (BR0); 0x1010A (BR1); 0x10112 (BR2); 0x1011A (BR3); 0x10122 (BR4); 0x1012A (BR5); 0x10132 (BR6); 0x1013A (BR7)									

[1] After a system reset, the valid bit is set in BR0 and cleared in BR1–BR7.

Figure 11-6. Base Registers (BR*x*)

Table 11-4 describes BRx fields.

Table 11-4. BR*x* Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base address 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 ORx[SDAM] for SDRAM and with ORx[AM] for GPCM and UPM.
17–18	—	Reserved, should be cleared
19–20	PS	Port size. Specifies the port size of this memory region. 01 8-bit 10 16-bit 11 32-bit 00 64-bit (60x bus only)
21–22	—	Reserved, should be cleared
23	WP	Write protect. Can restrict write accesses within the address range of a BR. An attempt to write to this address range while WP = 1 can cause $\overline{\text{TEA}}$ to be asserted by the bus monitor logic (if enabled) which terminates the cycle. 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert $\overline{\text{CSx}}$ and $\overline{\text{PSDVAL}}$ on write cycles to this memory bank. TESCR1[WP] is set if a write to this memory bank is attempted.

Table 11-4. BR*x* Field Descriptions (continued)

Bits	Name	Description
24–26	MS	Machine select. Specifies machine select for the memory operations handling and assigns the bank to the 60x if GPCM or SDRAM are selected. If UPM*x* is selected, the bus assignment is determined by M*x*MR[BSEL]. 000 GPCM—60x bus (reset value) 001 Reserved 010 SDRAM—60x bus 011 Reserved 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27	EMEMC	External MEMC enable. Overrides MS and assigns the bank to the 60x bus. However, other BRx fields remain in effect. See Section 11.2.7, “External Memory Controller Support.” 0 Access are handled by the memory controller according to MS. 1 Access are handled by an external memory controller (or other slave) on the 60x bus. The external memory controller is expected to assert $\overline{\text{AACK}}$, $\overline{\text{TA}}$, and $\overline{\text{PSDVAL}}$.
28–29	ATOM	Atomic operation. See Section 11.2.5, “Atomic Bus Operation.” 00 The address space controlled by the memory controller bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA).Writes to the address space handled by the memory controller bank cause the MPC8272 to lock the bus for the exclusive use of the master. The lock is released when the master performs a read operation from this address space. This feature is intended for CAM operations. 10 Write-after-read-atomic (WARA). Reads from the address space handled by the memory controller bank cause the MPC8272 to lock the bus for the exclusive use of the accessing device. The lock is released when the device performs a write operation to this address space. 11 Reserved **Note:** If the device fails to release the bus, the lock is released after 256 clock cycles.
30	DR	Data pipelining. See Section 11.2.6, “Data Pipelining.” 0 No data pipelining is done. 1 Data beats of accesses to the address space controlled by the memory controller bank are delayed by one cycle. This feature is intended for memory regions that need to improve data setup time.
31	V	Valid bit. Indicates that the contents of the BR*x* and OR*x* pair are valid. The $\overline{\text{CS}}$ signal does not assert until V is set. 0 This bank is invalid. 1 This bank is valid **Note:** An access to a region that has no V bit set may cause a bus monitor time-out. After a system reset, BR0[V] is set. **Note:** If BR*x* has been selected as the SDRAM controller and the valid bit has been set, the SDRAM controller must be invalidated by doing the following: 1. Disable the SDRAM refresh service by clearing PSDMR[RFEN]. 2. Wait at least 100 60x-bus clock cycles. 3. Clear BR*x*[V].

11.3.2 Option Registers (OR*x*)

The ORx registers define the sizes of memory banks and access attributes. The ORx attributes bits support the following three modes of operation as defined by BR[MS]:

- SDRAM mode
- GPCM mode
- UPM mode

Figure 11-7 shows the ORx as it is formatted for SDRAM mode.

	0	11	12	15
Field	SDAM		LSDAM...	
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x10104 (OR0); 0x1010C (OR1); 0x10114 (OR2); 0x1011C (OR3); 0x10124 (OR4); 0x1012C (OR5); 0x10134 (OR6); 0x1013C (OR7)			

	16	17	18	19	22	23	25	26	27	28	31
Field	...LSDAM	BPD		ROWST		NUMR		PMSEL	IBID	—	
Reset	0000_00000_0000_0000										
R/W	R/W										
Addr	0x10106 (OR0); 0x1010E (OR1); 0x10116 (OR2); 0x1011E (OR3); 0x10126 (OR4); 0x1012E (OR5); 0x10136 (OR6); 0x1013E (OR7)										

Figure 11-7. Option Registers (OR*x*)—SDRAM Mode

Table 11-5 describes ORx fields in SDRAM mode. For more details, see Section 11.4.12, “SDRAM Configuration Examples.”

Table 11-5. OR*x* Field Descriptions (SDRAM Mode)

Bits	Name	Description
0–11	SDAM	SDRAM address mask. Provides masking for corresponding BR*x* bits. By masking address bits independently, SDRAM devices of different size address ranges can be used. Clearing bits masks the corresponding address bit. Setting bits causes the corresponding address bit to be compared with the address pins. Address mask bits can be set or cleared in any order, allowing a resource to reside in more than one area of the address map. SDAM can be read or written at any time. 0000_0000_0000 4Gbytes 1000_0000_0000 2 Gbytes 1100_0000_0000 1 Gbyte 1110_0000_0000 512 Mbytes 1111_0000_0000 256 Mbytes 1111_1000_0000 128 Mbytes 1111_1100_0000 64 Mbytes 1111_1110_0000 32 Mbytes 1111_1111_0000 16 Mbytes 1111_1111_1000 8 Mbytes 1111_1111_1100 4 Mbytes 1111_1111_1110 2 Mbytes 1111_1111_1111 1 Mbyte **Note:** If xSDMR[PBI] = 0, the maximum size of the memory bank should not exceed 128 Mbytes.
12–16	LSDAM	Lower SDRAM address mask. Clearing LSDAM implements a minimum size of 1 Mbyte.
SDRAM Page Information		
17–18	BPD	Banks per device. Sets the number of internal banks per SDRAM device. 00 2 internal banks per device 01 4 internal banks per device 10 8 internal banks per device (not valid for 128-Mbyte SDRAMs) 11 Reserved **Note:** For 128-Mbyte SDRAMs, BPD must be 00 or 01.
19–22	ROWST	Row start address bit. Sets the demultiplexed row start address bit. The value of ROWST depends on SDMR[PBI]. For xSDMR[PBI] = 0: 0010 A7 0100 A8 0110 A9 1000 A10 1010 A11 1100 A12 1110 A13 Other values are reserved. For xSDMR[PBI] = 1: 0000 A0 0001 A1 ... 1100 A12 1101–1111 Reserved
23–25	NUMR	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 row address lines 001 10 row address lines 010 11 row address lines 011 12 row address lines 100 13 row address lines 101 14 row address lines 110 15 row address lines 111 16 row address lines

Table 11-5. OR*x* Field Descriptions (SDRAM Mode) (continued)

Bits	Name	Description
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	IBID	Internal bank interleaving within same device disable. Setting this bit disables bank interleaving between internal banks of a SDRAM device connected to the chip-select line. IBID should be set in 60x-compatible mode if the SDRAM device is not connected to the BANKSEL pins.
28–31	—	Reserved, should be cleared

Figure 11-8 shows ORx as it is formatted for GPCM mode.

	0–15
Field	AM
Reset[1]	1111_1110_0000_0000
R/W	R/W
Addr	0x10104 (OR0); 0x1010C (OR1); 0x10114 (OR2); 0x1011C (OR3); 0x10124 (OR4); 0x1012C (OR5); 0x10134 (OR6); 0x1013C (OR7)

	16	17–18	19	20	21–22	23	24–27	28	29	30	31
Field	AM	—	BCTLD	CSNT	ACS	—	SCY	SETA	TRLX	EHTR	—
Reset[1]	0	00	0	1	11	0	1111	0	1	0	0
R/W	R/W										
Addr	0x10106 (OR0); 0x1010E (OR1); 0x10116 (OR2); 0x1011E (OR3); 0x10126 (OR4); 0x1012E (OR5); 0x10136 (OR6); 0x1013E (OR7)										

Figure 11-8. OR*x* —GPCM Mode

Table 11-6 describes ORx fields in GPCM mode.

Table 11-6. OR*x*—GPCM Mode Field Descriptions

Bits	Name	Description
0–16	AM	Address mask. Masks corresponding BR*x* bits. Masking address bits independently allows external devices of different size address ranges to be used. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address pins. 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. **Note:** After system reset, OR0[AM] is 1111_1110_0000_0000_0.
17–18	—	Reserved, should be cleared
19	BCTLD	Data buffer control disable. Disables the assertion of $\overline{\mathrm{BCTL}x}$ during an access to the current memory bank. See Section 11.2.4, “Data Buffer Controls (BCTLx).” 0 $\overline{\mathrm{BCTL}x}$ are asserted upon an access to the current memory bank. 1 $\overline{\mathrm{BCTL}x}$ are not asserted upon an access to the current memory bank.

Table 11-6. ORx—GPCM Mode Field Descriptions (continued)

Bits	Name	Description
20	CSNT	Chip-select negation time. Determines when CS/WE are negated during an external memory write access handled by the GPCM. This helps meet address/data hold times for slow memories and peripherals. 0 CS/WE are negated normally. 1 CS/WE are negated a quarter of a clock earlier. (default) **Note:** After system reset OR0[CSNT] is set.
21–22	ACS	Address to chip select setup. Can be used when the external memory access is handled by the GPCM. It allows the delay of the CS assertion relative to the address change. 00 CS is output at the same time as the address lines 01 Reserved 10 CS is output a quarter of a clock after the address lines 11 CS is output half a clock after the address lines (default) **Note:** After a system reset, OR0[ACS] = 11.
23	—	Reserved, should be cleared
24–27	SCY	Cycle length in clocks. Determines the number of wait states inserted in the 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. The total memory access length is (2 + SCY) x Clocks. If the user selects an external PSDVAL response for this memory bank (by setting the SETA bit), write a non-zero values to SCY. 0000 = 0 clock cycle wait states...1111 = 15 clock cycles wait states **Note:** After a system reset, OR0[SCY] = 1111. **Note:** Refer to the note immediately following this table.
28	SETA	External access termination (PSDVAL generation). Used to specify that when the GPCM is selected to handle the memory access initiated to this memory region, the access is terminated externally by asserting the GTA external pin. In this case, PSDVAL is asserted one or two clocks later, depending on the synchronization of GTA. See Section 11.5.2, “External Access Termination.” 0 PSDVAL is generated internally by the memory controller unless GTA is asserted earlier externally. 1 PSDVAL is generated after external logic asserts GTA. **Note:** After a system reset, the OR0[SETA] is cleared.
29	TRLX	Timing relaxed. Works in conjunction with EHTR.
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 write access to the same bank, or any type of access to another bank. It does not affect subsequent read accesses to the same bank. TRLX and EHTR work together and are interpreted as follows: 00 Normal timing is generated by the memory controller. No additional cycles are inserted. 01 One idle clock cycle is inserted. 10 Four idle clock cycles are inserted. (default) 11 Eight idle clock cycles are inserted.
31	—	Reserved, should be cleared

NOTE

GPCM produces a glitch on the BSx lines when the following memory controller settings are used: SETA = 1, CSNT = 1, ACS = 01, TRLX = 1, and SCY = 0000.

During a write operation, the BSx are asserted for 3/4 of a cycle, negated for 1/4 of a cycle, and asserted again until the end of the cycle. Therefore, when the other conditions occur, it is necessary that SCY ≠ 0000.

Figure 11-9 shows ORx as it is formatted for UPM mode.

	0 … 15
Field	AM...
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x10104 (OR0); 0x1010C (OR1); 0x10114 (OR2); 0x1011C (OR3); 0x10124 (OR4); 0x1012C (OR5); 0x10134 (OR6); 0x1013C (OR7)

	16	17	18	19	20	22	23	24	28	29	30	31
Field	...AM	—		BCTLD	—		BI	—		EHTR		—
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x10106 (OR0); 0x1010E (OR1); 0x10116 (OR2); 0x1011E (OR3); 0x10126 (OR4); 0x1012E (OR5); 0x10136 (OR6); 0x1013E (OR7)											

Figure 11-9. OR*x*—UPM Mode

Table 11-7 describes the ORx fields in UPM mode.

Table 11-7. Option Register (OR*x*)—UPM Mode

Bits	Name	Description
0–16	AM	Address mask. Provides masking for corresponding BR*x* bits. By masking address bits independently, external devices of different size address ranges can be used. Any clear bit masks the corresponding address bit. Any set bit causes the corresponding address bit to be used in the comparison with the address pins. 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.
17–18	—	Reserved, should be cleared
19	BCTLD	Data buffer control disable. Disables the assertion of $\overline{\text{BCTL}x}$ during an access to the current memory bank. See Section 11.2.4, “Data Buffer Controls (BCTLx).” 0 $\overline{\text{BCTL}x}$ are asserted upon an access to the current memory bank. 1 $\overline{\text{BCTL}x}$ are not asserted upon an access to the current memory bank.
20–22	—	Reserved, should be cleared
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 UPMx executes burst accesses as series of single accesses.
24–28	—	Reserved, should be cleared

Table 11-7. Option Register (OR*x*)—UPM Mode (continued)

Bits	Name	Description
29–30	EHTR	Extended hold time on read accesses. Indicates how many cycles are inserted between a read access from the current bank and the next access. 00 Normal timing is generated by the memory controller. No additional cycles are inserted. 01 One idle clock cycle is inserted. 10 Four idle clock cycles are inserted. 11 Eight idle clock cycles are inserted.
31	—	Reserved, should be cleared

11.3.3 60x SDRAM Mode Register (PSDMR)

The 60x SDRAM mode register (PSDMR), shown in Figure 11-10, is used to configure operations pertaining to SDRAM.

	0	1	2 4	5 7	8	11 13	14 15
Field	PBI	RFEN	OP	SDAM	BSMA	SDA10	RFRC
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10190 (PSDMR)						

	16	17 19	20 22	23	24 25	26 27	28	29	30 31
Field	RFRC	PRETOACT	ACTTORW	BL	LDOTOPRE	WRC	EAMUX	BUFCMD	CL
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x10192 (PSDMR)								

Figure 11-10. 60x SDRAM Mode Register (PSDMR)

Table 11-8 describes PSDMR fields.

Table 11-8. PSDMR Field Descriptions

Bits	Name	Description
0	PBI	Page-based interleaving. Selects the address multiplexing method. PBI works in conjunction with PSDMR[SDA10]. See Section 11.4.5, “Bank Interleaving.” 0 Bank-based interleaving (default at reset) 1 Page-based interleaving (recommended operation)
1	RFEN	Refresh enable. Indicates that the SDRAM needs refresh services. 0 Refresh services are not required. 1 Refresh services are required. Note: After system reset, RFEN is cleared. See Section 11.3.7, “60x Bus-Assigned UPM Refresh Timer (PURT),” Section 11.3.8, “60x Bus-Assigned SDRAM Refresh Timer (PSRT).”

Table 11-8. PSDMR Field Descriptions (continued)

Bits	Name	Description
2–4	OP	SDRAM operation. Determines which operation occurs when the SDRAM device is accessed. 000 Normal operation 001 CBR refresh, used in SDRAM initialization 010 Self refresh (for debug purpose) 011 Mode register write, used in SDRAM initialization Note that if 60x-compatible mode is in effect on the 60x bus or the SDRAM port size is 8/16 or the SDRAM is connected to the BADDR lines (not needed for 64/32 port size), the bus master must supply the mode register data on the low bits of the address during the access. 100 Precharge bank (for debug purpose) 101 Precharge all banks, used in SDRAM initialization 110 Activate bank (for debug purpose) 111 Read/write (for debug purpose)
5–7	SDAM	Address multiplex size. Determines how the address of the current memory cycle can be output on the address pins. See Section 11.4.5.2, “SDRAM Address Multiplexing (SDAM and BSMA).”
8–10	BSMA	Bank select multiplexed address line. Selects the address pins to serve as bank-select address for the 60x SDRAM. The bank select address can also be output on the BANKSEL pins (optional). See Section 11.4.5.2, “SDRAM Address Multiplexing (SDAM and BSMA).” 000 A12:A14 001 A13:A15 010 A14:A16 011 A15:A17 100 A16:A18 101 A17:A19 110 A18:A20 111 A19:A21
11–13	SDA10	'A10' control. With PSDMR[PBI], determines which address line can be output to SDA10 during an ACTIVATE command, when SDRAM is selected, to control the memory access. See Section 11.4.12.1, “SDRAM Configuration Example (Page-Based Interleaving).” **For PBI = 0:** 000 A12 001 A11 010 A10 011 A9 100 A8 101 A7 110 A6 111 A5 **For PBI = 1:** 000 A10 001 A9 010 A8 011 A7 100 A6 101 A5 110 A4 111 A3
SDRAM Device–Specific Parameters:		
14–16	RFRC	Refresh recovery. Defines the earliest timing for an activate command after a REFRESH command. Sets the refresh recovery interval in clock cycles. See Section 11.4.6.6, “Refresh Recovery Interval (RFRC)v for how to set this field. 000 Reserved 001 3 clocks 010 4 clocks 011 5 clocks 100 6 clocks 101 7 clocks 110 8 clocks 111 16 clocks

Table 11-8. PSDMR Field Descriptions (continued)

Bits	Name	Description
17–19	PRETOACT	Precharge to activate interval. Defines the earliest timing for ACTIVATE or REFRESH command after a precharge command. See Section 11.4.6.1, “Precharge-to-Activate Interval.” 001 1 clock-cycle wait states 010 2 clock-cycle wait states ... 111 7 clock-cycle wait states 000 8 clock-cycle wait states
20–22	ACTTORW	Activate to read/write interval. Defines the earliest timing for READ/WRITE command after an ACTIVATE command. See Section 11.4.6.2, “Activate to Read/Write Interval.” 001 1 clock cycle 010 2 clock cycles ... 111 7 clock cycles 000 8 clock cycles
23	BL	Burst length 0 SDRAM burst length is 4. Use this value if the device port size is 64 or 16 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8
24–25	LDOTOPRE	Last data out to precharge. Defines the earliest timing for PRECHARGE command after the last data was read from the SDRAM. See Section 11.4.6.4, “Last Data Out to Precharge.” 00 0 clock cycles 01 1 clock cycle 10 2 clock cycles 11 Reserved **Note:** A value of 0b00 (0 clock cycles) gives the longest time while the a value of 0b10 gives the least.
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data was written to the SDRAM. See Section 11.4.6.5, “Last Data In to Precharge—Write Recovery.” 01 1 clock cycles 10 2 clock cycles 11 3 clock cycles 00 4 clock cycles
28	EAMUX	External address multiplexing enable/disable. 0 No external address multiplexing. Fastest timing. 1 The memory controller asserts SDAMUX for an extra cycle before issuing an ACTIVATE command to the SDRAM. This is useful when external address multiplexing can cause a delay on the address lines. Note that if this bit is set, ACTTORW should be a minimum of 2. In 60x-compatible mode, external address multiplexing is placed on the address lines. If the additional delay of the multiplexing endangers the device setup time, EAMUX should be set. Setting this bit causes the memory controller to add another cycle for each address phase. Note that EAMUX can also be set in any case of delays on the address lines, such as address buffers. See Section 11.4.6.7, “External Address Multiplexing Signal.”

Table 11-8. PSDMR Field Descriptions (continued)

Bits	Name	Description
29	BUFCMD	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 $\overline{CS}$ to be asserted for two cycles, instead of one. See Section 11.4.6.8, “External Address and Command Buffers (BUFCMD).” 0 Normal timing for the control lines 1 All control lines except $\overline{CS}$ are asserted for two cycles In 60x-compatible mode, external buffers may be placed on the command strobes, except $\overline{CS}$, as well as the address lines. If the additional delay of the buffers endangers the device setup time, BUFCMD should be set, which causes the memory controller to add a cycle for each SDRAM command.
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. See Section 11.4.6.3, “Column Address to First Data Out—CAS Latency.” 00 Reserved 01 1 10 2 11 3

11.3.4 Machine A/B/C Mode Registers (M*x*MR)

The machine x mode registers (MxMR), shown in Figure 11-11, contain the configuration for the three UPMs.

	0	1	2 3	4	5 7	8 9	10 12	13	14 15
Field	—	RFEN	OP	—	AM*x*	DSx	G0CLx	GPL_x4DIS	RLFx
Reset	0000_0000_0000_0							1	00
R/W	R/W								
Addr	0x10170 (MAMR); 0x10174 (MBMR); 0x10178 (MCMR)								

	16 17	18 21	22 25	26 31
Field	RLFx	WLF*x*	TLF*x*	MAD
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x10172 (MAMR); 0x10176 (MBMR); 0x1017A (MCMR)			

Figure 11-11. Machine *x* Mode Registers (M*x*MR)

Table 11-9 describes MxMR bits.

Table 11-9. Machine x Mode Registers (M*x*MR)

Bits	Name	Description
0	—	Reserved, should be cleared
1	RFEN	Refresh enable. Indicates that the UPM needs refresh services. 0 Refresh services are not required. 1 Refresh services are required. See Section 11.3.7, “60x Bus-Assigned UPM Refresh Timer (PURT),” and Section 11.3.8, “60x Bus-Assigned SDRAM Refresh Timer (PSRT).”
2–3	OP	Command opcode. Determines the command executed by the UPMx when a memory access hit a UPM assigned bank. 00 Normal operation 01 Write to array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed by MAD. After the access, the MAD field is automatically incremented. 10 Read from array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed by MAD into the MDR. After the access, the MAD field 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 by MAD and continues until the LAST bit is set in the RAM. **Note:** RLF determines the number of times a loop is executed during a pattern run.
4	—	Reserved, should be cleared
5–7	AMx	Address multiplex size. Determines how the address of the current memory cycle can be output on the address pins. The address output on the pins controlled by the contents of the UPM*x* RAM array. This field is useful when connecting the MPC8272 to DRAM devices requiring row and column addresses multiplexed on the same pins. See Section 11.6.4.2, “Address Multiplexing.”
8–9	DSx	Disable timer period. Guarantees a minimum time between accesses to the same memory bank if it is controlled by the UPMx. The disable timer is turned on by the TODT in the RAM array, and when expired, the UPMx allows the machine access to handle a memory pattern to the same memory region. Accesses to a different memory region by the same UPMx will be allowed. 00 1-cycle disable period 01 2-cycle disable period 10 3-cycle disable period 11 4-cycle disable period **Note:** To avoid conflicts between successive accesses to different memory regions, the minimum pattern in the RAM array for a request serviced should not be shorter than the period established by DS*x*.
10–12	G0CLx	General line 0 control. Determines which address line can be output to the GPL0 pin when the UPMx is selected to control the memory access. 000 A12 001 A11 010 A10 011 A9 100 A8 101 A7 110 A6 111 A5

Table 11-9. Machine x Mode Registers (M*x*MR) (continued)

Bits	Name	Description
13	GPL_x4DIS	GPL_A4 output line disable. Determines if the UPMWAIT/$\overline{\text{GTA}}$/GPL_4 pin behaves as an output line controlled by the corresponding bits in the UPMx array (GPL4x). 0 UPMWAIT/$\overline{\text{GTA}}$/GPL_x4 behaves as GPL_4. UPMx[G4T4/DLT3] is interpreted as G4T4. The UPMx[G4T3/WAEN] is interpreted as G4T3. 1 UPMWAIT/$\overline{\text{GTA}}$/GPL_x4 behaves as UPMWAIT. UPMx[G4T4/DLT3] is interpreted as DLT3. UPMx[G4T3/WAEN] is interpreted as WAEN. **Note:** After a system reset, GPL_x4DIS = 1.
14–17	RLFx	Read loop field. Determines the number of times a loop defined in the UPMx will be executed for a burst- or single-beat read pattern or when M*x*MR[OP] = 11 (RUN command) 0001 The loop is executed 1 time. 0010 The loop is executed 2 times. ... 1111 The loop is executed 15 times. 0000 The loop is executed 16 times.
18–21	WLFx	Write loop field. Determines the number of times a loop defined in the UPMx will be executed for a burst- or single-beat write pattern. 0001 The loop is executed 1 time. 0010 The loop is executed 2 times. ... 1111 The loop is executed 15 times. 0000 The loop is executed 16 times.
22–25	TLFx	Refresh loop field. Determines the number of times a loop defined in the UPMx will be executed for a refresh service pattern. 0001 The loop is executed 1 time. 0010 The loop is executed 2 times. ... 1111 The loop is executed 15 times. 0000 The loop is executed 16 times.
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.

11.3.5 Memory Data Register (MDR)

The memory data register (MDR), shown in Figure 11-12, 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.

	0 … 15
Field	MD
Reset	xxxx_xxxx_xxxx_xxxx[1]
R/W	R/W
Addr	0x10188

	16 … 31
Field	MD
Reset	xxxx_xxxx_xxxx_xxxx[1]
R/W	R/W
Addr	0x1018A

[1] Undefined at reset.

Figure 11-12. Memory Data Register (MDR)

Table 11-10 describes MDR fields.

Table 11-10. MDR Field Descriptions

Bits	Name	Description
0–31	MD	Memory data. The data to be read or written into the RAM array when a WRITE or READ command is supplied to the UPM.

11.3.6 Memory Address Register (MAR)

The memory address register (MAR) is shown in Figure 11-13.

	0 … 15
Field	A
Reset	xxxx_xxxx_xxxx_xxxx[1]
R/W	R/W
Addr	0x10168

	16 … 31
Field	A
Reset	xxxx_xxxx_xxxx_xxxx[1]
R/W	R/W
Addr	0x10116A

[1] Undefined at reset.

Figure 11-13. Memory Address Register (MAR)

Table 11-11 describes MAR fields.

Table 11-11. MAR Field Description

Bits	Name	Description
0–31	A	Memory address. The memory address register can be output to the address lines under control of the AMX bits in the UPM

11.3.7 60x Bus-Assigned UPM Refresh Timer (PURT)

The 60x bus assigned UPM refresh timer register (PURT) is shown in Figure 11-14.

	0 — 7
Field	PURT
Reset	0000_0000
R/W	R/W
Addr	0x10198

Figure 11-14. 60x Bus-Assigned UPM Refresh Timer (PURT)

Table 11-12 describes PURT fields.

Table 11-12. 60x Bus-Assigned UPM Refresh Timer (PURT)

Bits	Name	Description
0–7	PURT	Refresh timer period. Determines the timer period according to the following equation: $\text{TimerPeriod} = \left(\frac{(\text{PURT} + 1) \times (\text{MPTPR[PTP]} + 1)}{\text{Bus Frequency}}\right)$ This timer generates a refresh request for all valid banks that selected a UPM machine assigned to the 60x bus (M*x*MR[BSEL] = 0) and is refresh-enabled (M*x*MR[RFEN] = 1). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks are rotating their requests. Example: For a 25-MHz system clock and a required service rate of 15.6 µs, given MPTPR[PTP] = 31, the PURT value should be 11decimal. (12*32)/25 MHz = 15.36 µs, which is less than the required service period of 15.6 µs.

11.3.8 60x Bus-Assigned SDRAM Refresh Timer (PSRT)

The 60x bus assigned SDRAM refresh timer register (PSRT) is shown in Figure 11-15.

	0 — 7
Field	PSRT
Reset	0000_0000
R/W	R/W
Addr	0x1019C

Figure 11-15. 60x Bus-Assigned SDRAM Refresh Timer (PSRT)

Table 11-13 describes PSRT fields.

Table 11-13. 60x Bus-Assigned SDRAM Refresh Timer (PSRT)

Bits	Name	Description
0–7	PSRT	Refresh timer period. Determines the timer period according to the following equation: $\text{TimerPeriod} = \left(\frac{(\text{PSRT} + 1) \times (\text{MPTPR[PTP]} + 1)}{\text{Bus Frequency}}\right)$ This timer generates refresh requests for all valid banks that selected a SDRAM machine assigned to the 60x bus and is refresh-enabled (PSDMR[RFEN] = 1). Each time the timer expires, all banks that qualify generate a bank staggering auto refresh request using the SDRAM machine. See Section 11.4.10, “SDRAM Refresh.” Example: For a 25-MHz system clock and a required service rate of 15.6 µs, given MPTPR[PTP] = 31, the PSRT value should be 11decimal. (12 × 32)/25 MHz = 15.36 µs, which is less than the required service period of 15.6 µs.

11.3.9 Memory Refresh Timer Prescaler Register (MPTPR)

Figure 11-16 shows the memory refresh timer prescaler register (MPTPR).

	0	7	8	15
Field	PTP		—	
Reset	Undefined			
R/W	R/W			
Addr	0x10184			

Figure 11-16. Memory Refresh Timer Prescaler Register (MPTPR)

Table 11-14 describes MPTPR fields.

Table 11-14. MPTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the memory refresh timers input clock. It divides the *bus* clock. Prescaler clock frequency = Bus frequency / (PTP + 1).
8–15	—	Reserved, should be cleared

11.3.10 60x Bus Error Status and Control Registers (TESCR*x*)

These registers indicate the source of an error that caused $\overline{\text{TEA}}$ or $\overline{\text{MCP}}$ to be asserted on the 60x bus. See Section 4.3.2.8, “60x Bus Transfer Error Status and Control Register 1 (TESCR1),” and Section 4.3.2.9, “60x Bus Transfer Error Status and Control Register 2 (TESCR2).”

11.4 SDRAM Machine

The MPC8272 provides one SDRAM interface (machine) for the 60x bus. The machines provide the necessary control functions and signals for JEDEC-compliant SDRAM devices.

Each bank can control a SDRAM device on the 60x. Table 11-15 describes the SDRAM interface signals controlled by the memory controller.

Table 11-15. SDRAM Interface Signals

60x Bus	Comments
$\overline{\text{CS}}$[0:7]	Device select
PSDRAS	RAS
SDCAS	CAS
SDWE	WEN
SDA10	'A10' control
$\overline{\text{DQM}}$[0:7]	Byte select

Additional controls are available in 60x-compatible mode (60x bus only):

- ALE—External address latch enable
- PSDAMUX—External address multiplexing control (asserted = row, negated = column)

Throughout this section, whenever a signal is named, the reference is to the 60x signal name.

Figure 11-17 shows an eight-bank, 128-Mbyte system. Each bank consists of eight 2 x 1-Mbit x 8 SDRAMs. Note that the SDRAM memory clock must operate at the same frequency as, and be phase-aligned with, the system clock.

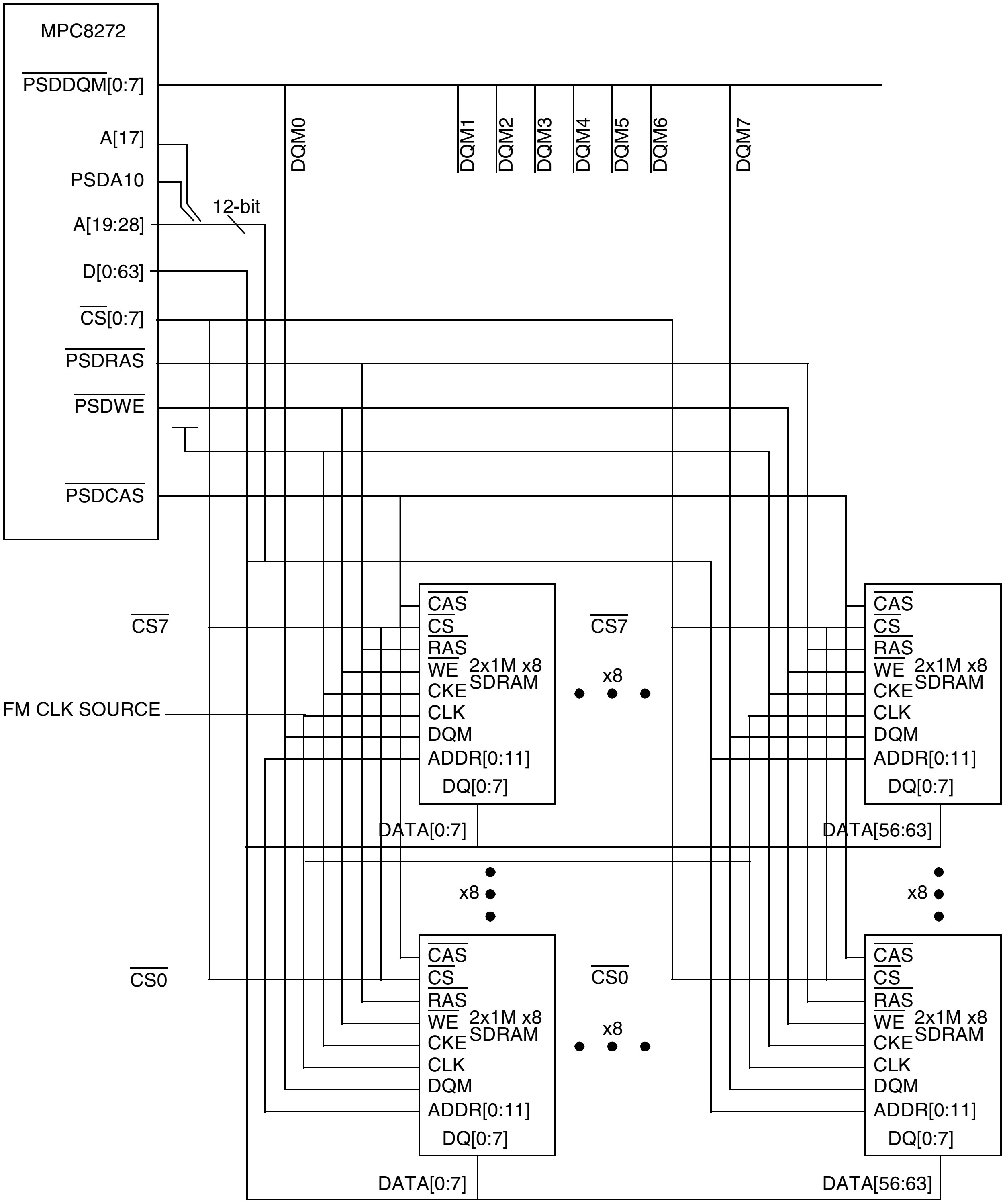


Figure 11-17. 128-Mbyte SDRAM (Eight-Bank Configuration, Banks 1 and 8 Shown)

11.4.1 Supported SDRAM Configurations

The MPC8272 memory controller supports any SDRAM configuration under the restrictions that all SDRAM devices should have the same port size and timing parameters.

11.4.2 SDRAM Power-On Initialization

At system reset, initialization software must set up the programmable parameters in the memory controller banks registers (ORx, BRx, P/LSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

1. Issue a PRECHARGE-ALL-BANKS command.
2. Issue eight CBR REFRESH commands. If the SDRAM does not require eight CBR REFRESH commands, then the SDRAM requirement should be followed.
3. Issue a MODE-SET command to initialize the mode register.

The initial commands are executed by setting P/LSDMR[OP] and accessing the SDRAM with a single-byte transaction. See Figure 11-10.

Note that software should ensure that no memory operations begin until this process completes.

11.4.3 JEDEC-Standard SDRAM Interface Commands

The MPC8272 performs all accesses to SDRAM by using JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the MPC8272 bus clock. Data at the output of the SDRAM device must be sampled on the rising edge of the MPC8272 bus clock.

As seen in Table 11-16, the MPC8272 provides the following SDRAM interface commands:

Table 11-16. SDRAM Interface Commands

Command	Description
BANK-ACTIVATE	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 BANK-ACTIVATE is issued.
MODE-SET	Allows setting of SDRAM options—$\overline{\text{CAS}}$ latency, burst type, and burst length. $\overline{\text{CAS}}$ latency depends on the SDRAM device used (some SDRAMs provide CAS latency of 1, 2, or 3; some provide a latency of 1, 2, 3, or 4, etc.). Burst type must be chosen according to the 60x cache wrap (sequential). Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, MPC8272 supports only a 4-beat burst for 64-bit port size and an 8-beat burst for 32-bit port size. MPC8272 does not support burst lengths of 1, 2, and a page for SDRAMs. The mode register data (CAS latency, burst length, and burst type) is programmed into the P/LSDMR register by initialization software at reset. After the P/LSDMR is set, the MPC8272 transfers the information to the SDRAM array by issuing a MODE-SET command. Section 11.4.9, “SDRAM Mode-Set Command Timing,” gives timing information.
PRECHARGE (SINGLE BANK/ALL BANKS)	Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers to prepare for reading another row in the SDRAM array. A PRECHARGE command must be issued after a read or write if the row address changes on the next access. Note that the MPC8272 uses the SDA10 pin to distinguish the PRECHARGE-ALL-BANKS command. The SDRAMs must be compatible with this format.

Table 11-16. SDRAM Interface Commands (continued)

Command	Description
READ	Latches the column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each successive clock, additional data is output without additional READ commands. The amount of data transferred is determined by the burst size. At the end of the burst, the page remains open.
REFRESH	Causes a row to be read in both 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. Both banks must be in a precharged state before executing REFRESH.
WRITE	Latches the column address and transfers data from the data signals to the selected sense amplifier 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. The amount of data transferred is determined by the burst size. At the end of the burst, the page remains open.

11.4.4 Page-Mode Support and Pipeline Accesses

The SDRAM interface supports back-to-back page mode. A page remains open as long as back-to-back accesses that hit the page are generated on the bus. The page is closed once the bus becomes idle unless ORx[PMSEL] is set.

The use of SDRAM pipelining allows data phases to occur on with zero bubbles for CPM accesses and with one bubble for core accesses, as required by the 60x bus specification.

If ETM/LETM = 1, the use of SDRAM pipelining also allows for back-to-back data phases to occur with zero clocks of separation for CPM accesses and with one clock of separation for core accesses, as required by the 60x bus specification.

11.4.5 Bank Interleaving

The SDRAM interface supports bank interleaving. This means that if a missed page is in a different SDRAM bank than the currently open page, the SDRAM machine first issues an ACTIVATE command to the new page and later issues a DEACTIVATE command to the old page, thus eliminating the DEACTIVATE time overhead.

This procedure can be done if both pages reside on different SDRAM devices or on different internal SDRAM banks. The second option can be disabled by setting ORx[IBID]. The user should set this bit if the BNKSEL pins are not used in 60x-compatible mode.

The following two methods are used for internal bank interleaving:

- Page-based interleaving—Yields the best performance and is the preferred interleaving method. This method uses low address bits as the bank-select for the SDRAM, thus allowing interleaving on every page boundary. It is activated by setting xSDMR[PBI]=1. See “0xSDRAM Configuration Example (Page-Based Interleaving)”.
- Bank-based interleaving —This method uses the most-significant address bits as the bank-select for the SDRAM, thus allowing interleaving only on bank boundaries. It is activated by clearing xSDMR[PBI]. See Section 11.4.12, “SDRAM Configuration Examples.”

11.4.5.1 Using BNKSEL Signals in Single-MPC8272 Bus Mode

The BNKSEL signals provide the following functionality in single-MPC8272 bus mode

- If bank-based interleaving is used, BNKSEL signals facilitate compatibility with SDRAMs that have different numbers of row or column address lines. The address lines of the MPC8272 bus and the BNKSEL lines can be routed independently to the address lines and BA lines of the DIMM. Note that all SDRAMs populated on an MPC8272 bus must still have the same organization. This flexibility merely allows the SDRAMs to be populated as a group with larger or smaller devices as appropriate.

  If BNKSEL lines were not used, the number of row and column address lines of the SDRAMs would affect which MPC8272 address bus lines on which the bank select signals would be driven, and would thus require that the BA signals of the SDRAMs be routed to those address lines, thus limiting flexibility.
- If BCR[EAV] is programmed, BNKSEL signals facilitate logic analysis of the system. Otherwise, the logic analyzer equipment must understand the address multiplexing scheme of the board and intelligently reconstruct the address of bus transactions.

11.4.5.2 SDRAM Address Multiplexing (SDAM and BSMA)

In single MPC8272 mode, the lower bits of the address bus are connected to the device's address port, and the memory controller multiplex the row/column and the internal banks select lines, according to the PL/SDMR[SDAM] and PL/SDMR[BSMA].

Table 11-17 shows how P/LSDMR[SDAM] settings affect address multiplexing. For the effect of PL/SDMR[BSMA] see Section 11.4.12, "SDRAM Configuration Examples."

Note that in 60x-compatible mode, the 60x address must be latched and multiplexed by glue logic that is controlled by ALE and SDAMUX, however, the user still has to configure PSDMR[SDAM].

Table 11-17 shows SDRAM address multiplexing for A0:A15.

Table 11-17. SDRAM Address Multiplexing (A0:A15)

SDAM	External Bus Address Pins	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
000	Signal driven on external pins when address multiplexing is enabled	—	—	—	—	—	—	—	—	—	—	—	—	—	A5	A6	A7
001		—	—	—	—	—	—	—	—	—	—	—	—	—	—	A5	A6
010		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	A5
011		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
100		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
101		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 11-18 shows SDRAM address multiplexing for A16:A31.

Table 11-18. SDRAM Address Multiplexing (A16:A31)

SDAM	External Bus Address Pins	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external pins when address multiplexing is enabled	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		—	—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

11.4.6 SDRAM Device-Specific Parameters

The software is responsible for setting correct values to some device-specific parameter that can be extracted from the data sheet. The values are stored in the ORx and P/LSDMR registers. These parameters include the following:

- Precharge to activate interval (P/LSDMR[PRETOACT]). See Section 11.4.6.1, "Precharge-to-Activate Interval."
- Activate to read/write interval (P/LSDMR[ACTTORW]). See Section 11.4.6.2, "Activate to Read/Write Interval."
- CAS latency, column address to first data out (P/LSDMR[CL]). See Section 11.4.6.3, "Column Address to First Data Out—CAS Latency."
- Last data out to precharge (P/LSDMR[LDOTOPRE]). Section 11.4.6.4, "Last Data Out to Precharge."
- Write recovery, last data in to precharge (P/LSDMR[WRC]). See Section 11.4.6.5, "Last Data In to Precharge—Write Recovery."
- Refresh recovery interval (P/LSDMR[RFRC]). See Section 11.4.6.6, "Refresh Recovery Interval (RFRC)."
- External address multiplexing present (P/LSDMR[EAMUX]). See Section 11.4.6.7, "External Address Multiplexing Signal."
- External buffers on the control lines present (P/LSDMR[BUFCMD]). See Section 11.4.6.8, "External Address and Command Buffers (BUFCMD)."

The following sections describe the SDRAM parameters that are programmed in the P/LSDMR register.

11.4.6.1 Precharge-to-Activate Interval

As demonstrated in Figure 11-18, this parameter, controlled by P/LSDMR[PRETOACT] defines the earliest timing for activate or refresh command after a precharge command.

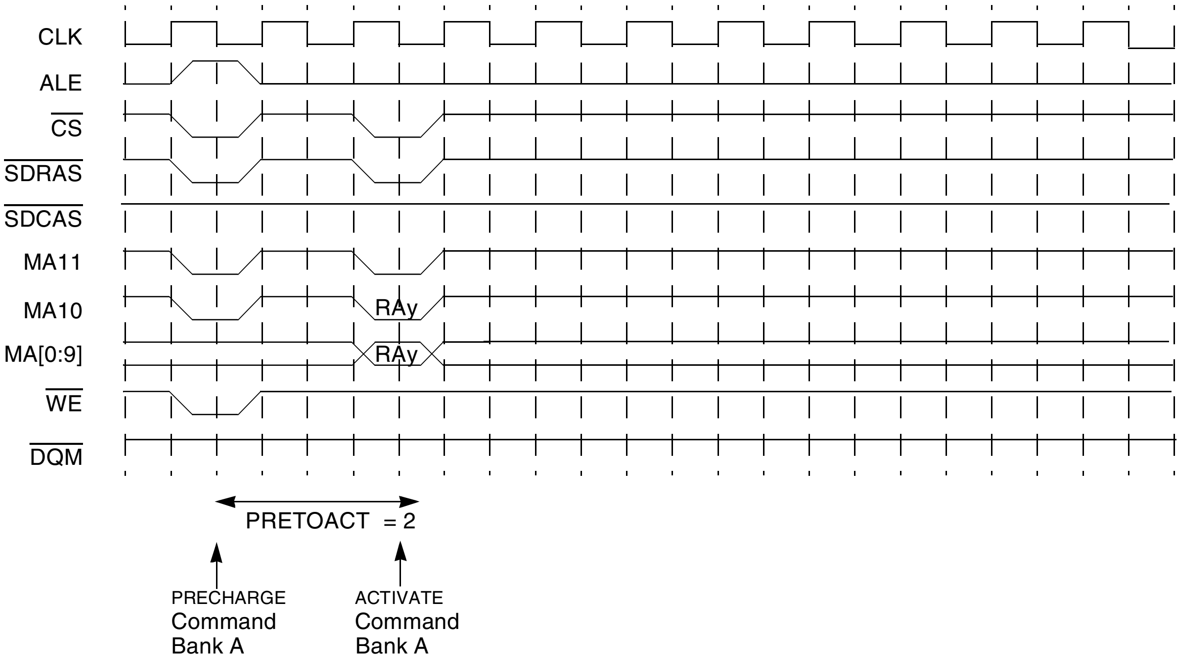


Figure 11-18. PRETOACT = 2 (2 Clock Cycles)

11.4.6.2 Activate to Read/Write Interval

As represented in Figure 11-19, this parameter, controlled by P/LSDMR[ACTTORW], defines the earliest timing for READ/WRITE command after an ACTIVATE command.

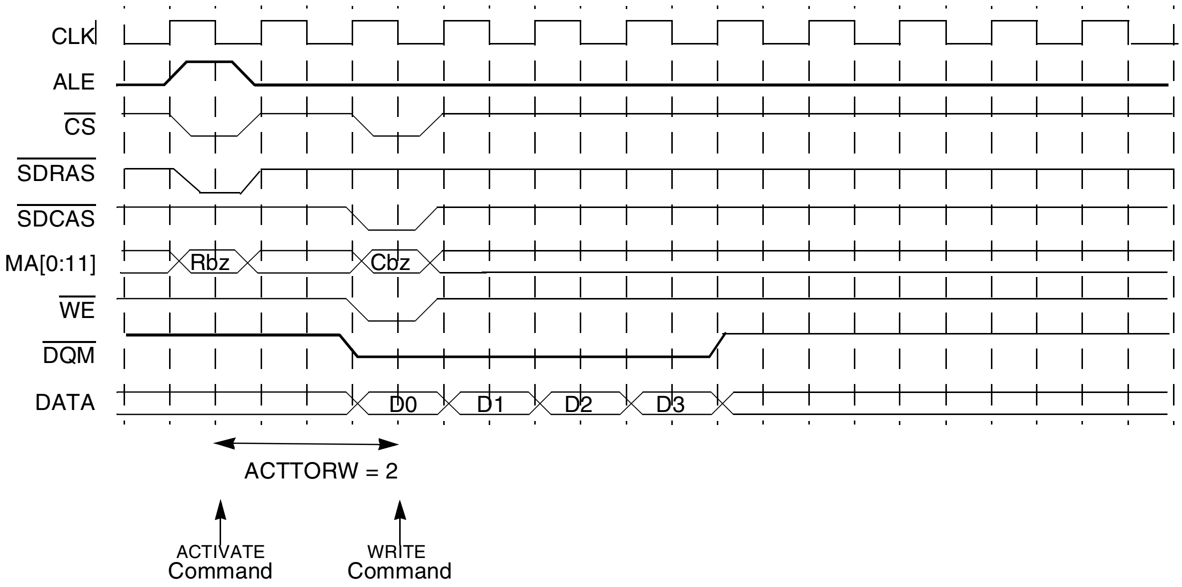


Figure 11-19. ACTTORW = 2 (2 Clock Cycles)

11.4.6.3 Column Address to First Data Out—CAS Latency

As seen in Figure 11-20, this parameter, controlled by P/LSDMR[CL], defines the timing for first read data after a column address is sampled by the SDRAM.

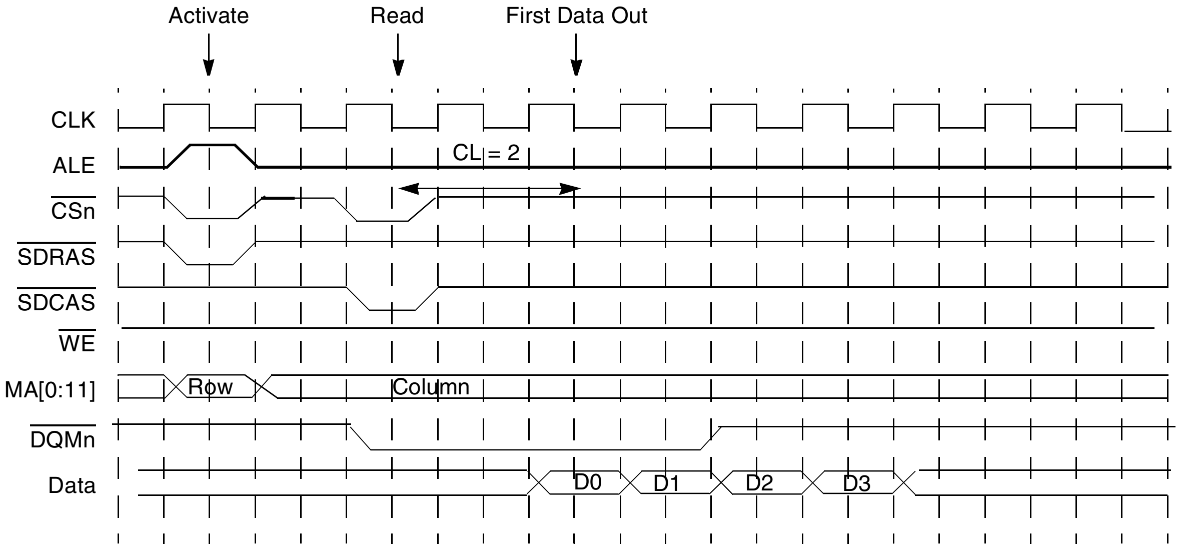


Figure 11-20. CL = 2 (2 Clock Cycles)

11.4.6.4 Last Data Out to Precharge

As shown in Figure 11-21, this parameter, controlled by P/LSDMR[LDOTOPRE], defines the earliest timing for the PRECHARGE command after the last data was read from the SDRAM. It is always related to the CL parameter.

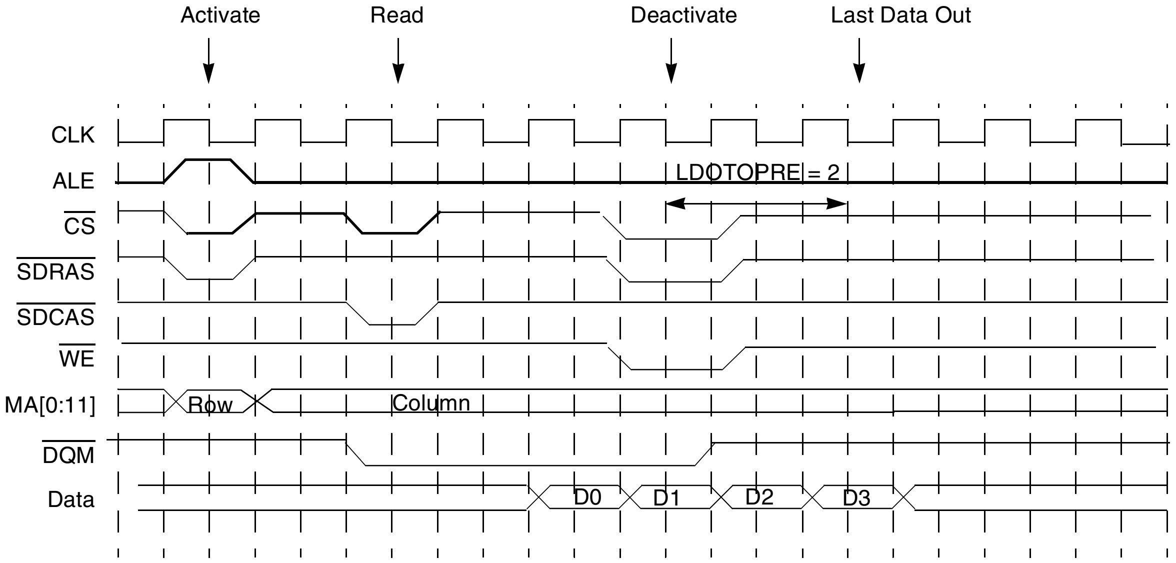


Figure 11-21. LDOTOPRE = 2 (2 Clock Cycles)

11.4.6.5 Last Data In to Precharge—Write Recovery

As demonstrated in Figure 11-22, this parameter, controlled by P/LSDMR[WRC], defines the earliest timing for PRECHARGE command after the last data was written to the SDRAM.

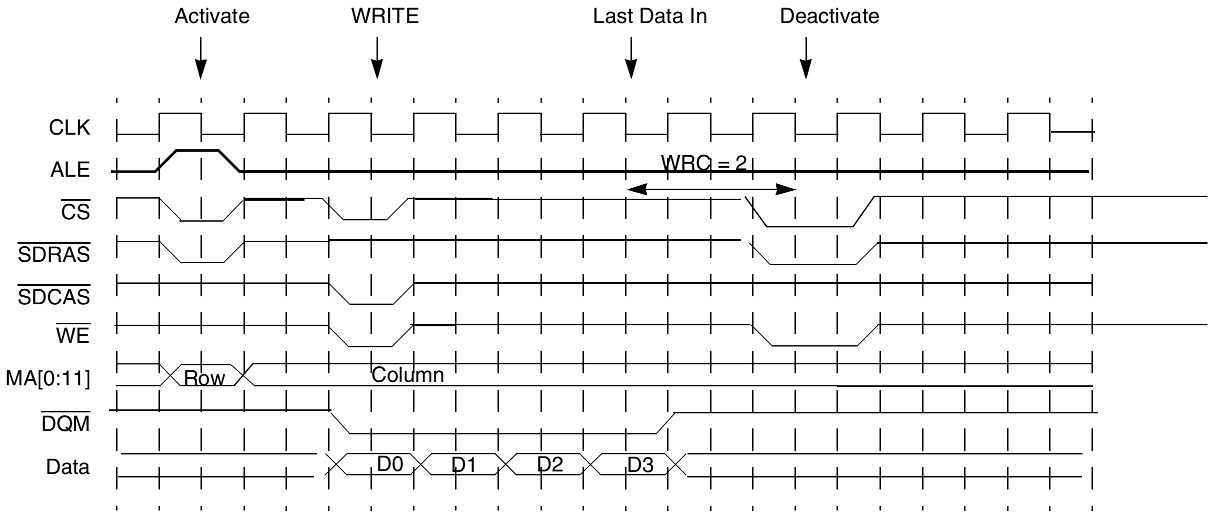


Figure 11-22. WRC = 2 (2 Clock Cycles)

11.4.6.6 Refresh Recovery Interval (RFRC)

As represented in Figure 11-23, this parameter, controlled by P/LSDMR[RFRC], defines the earliest timing for an ACTIVATE command after a REFRESH command.

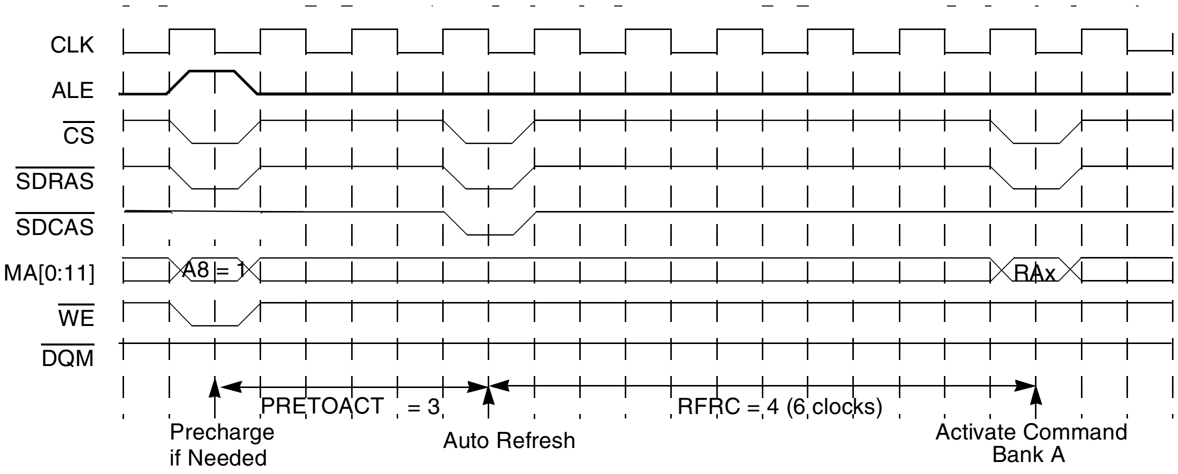


Figure 11-23. RFRC = 4 (6 Clock Cycles)

11.4.6.7 External Address Multiplexing Signal

In 60x-compatible mode, external address multiplexing is placed on the address lines. If the additional delay of multiplexing endangers the device setup time, P/LSDMR[EAMUX] should be set. Setting this bit causes the memory controller to add another cycle for each address phase. Figure 11-24 demonstrates the timing when EAMUX equals 1.

Note that EAMUX can also be set in any case of delays on the address lines, such as address buffers.

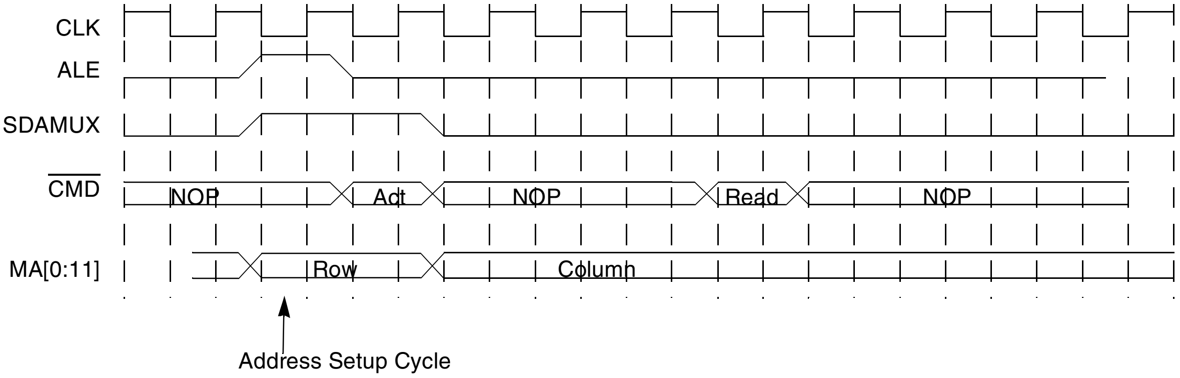


Figure 11-24. EAMUX = 1

11.4.6.8 External Address and Command Buffers (BUFCMD)

In 60x-compatible mode, external buffers may be placed on the command strobes, except $\overline{CS}$, as well as the address lines. If the additional delay of the buffers is endangering the device setup time, P/LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add one cycle for each SDRAM command. Figure 11-25 illustrates the timing when BUFCMD equals 1.

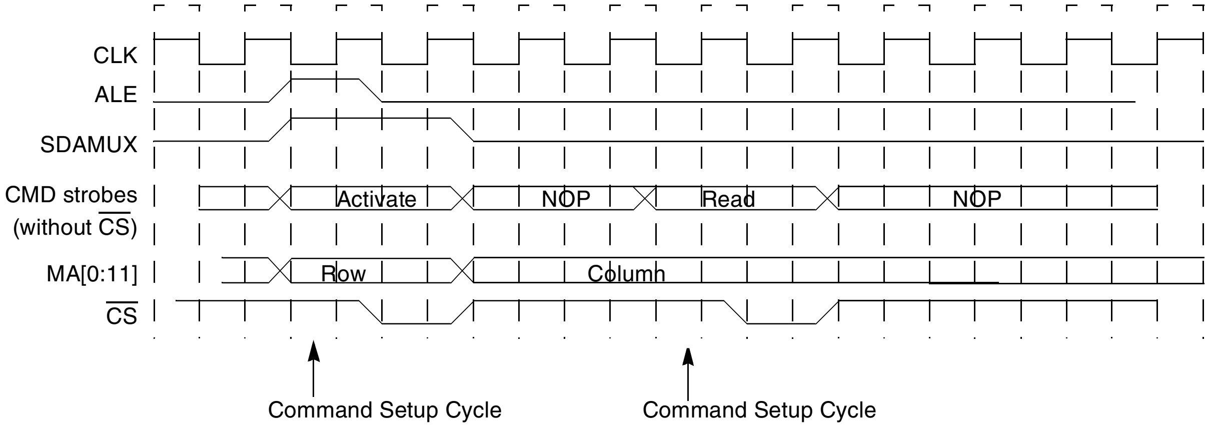


Figure 11-25. BUFCMD = 1

11.4.7 SDRAM Interface Timing

Figure 11-26 through Figure 11-34 show SDRAM timing for various types of accesses.

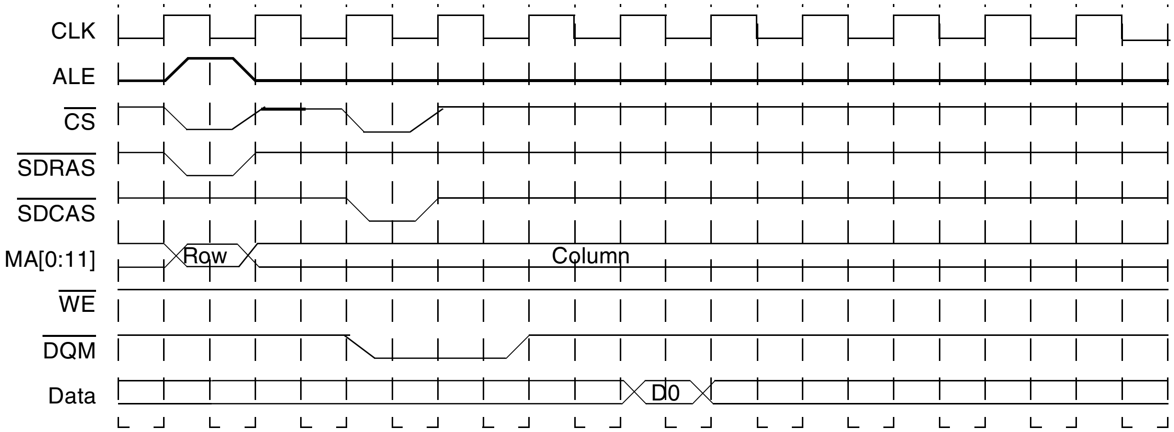


Figure 11-26. SDRAM Single-Beat Read, Page Closed, CL = 3

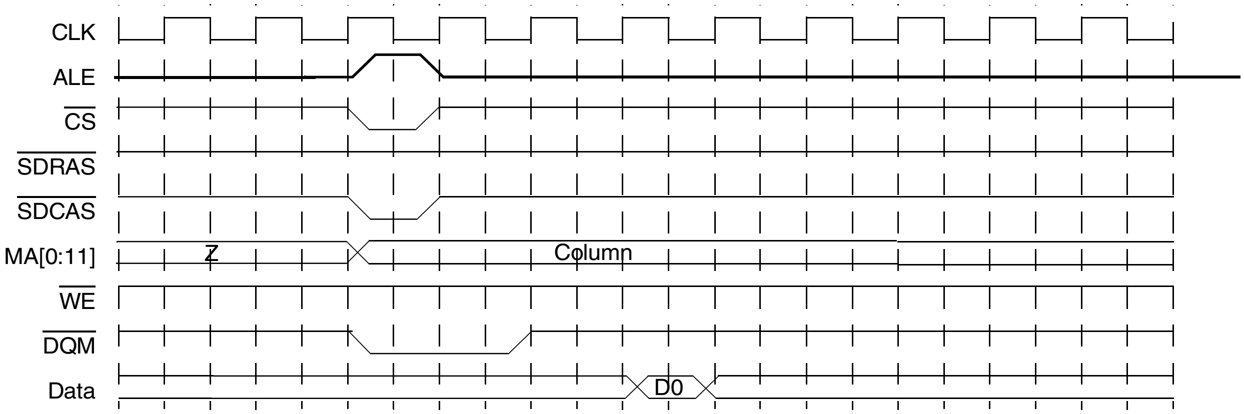


Figure 11-27. SDRAM Single-Beat Read, Page Hit, CL = 3

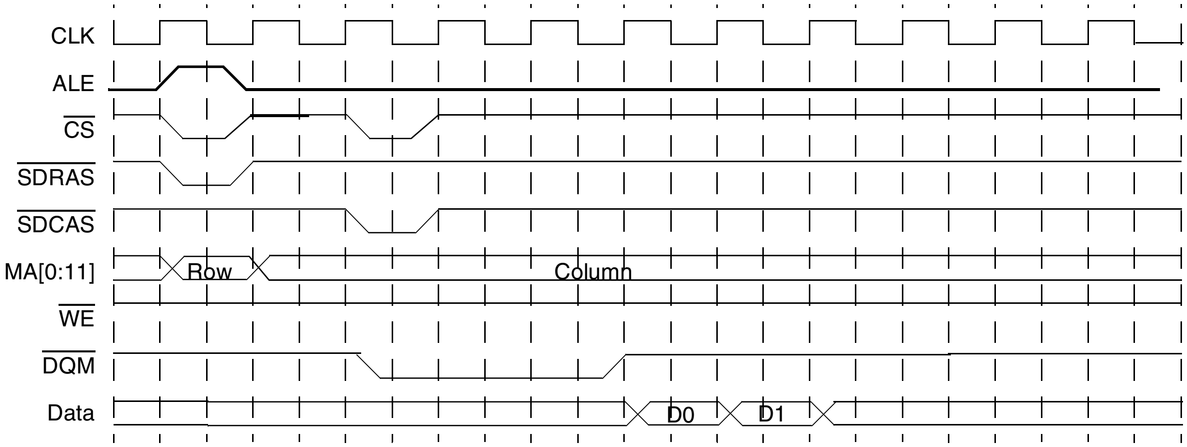
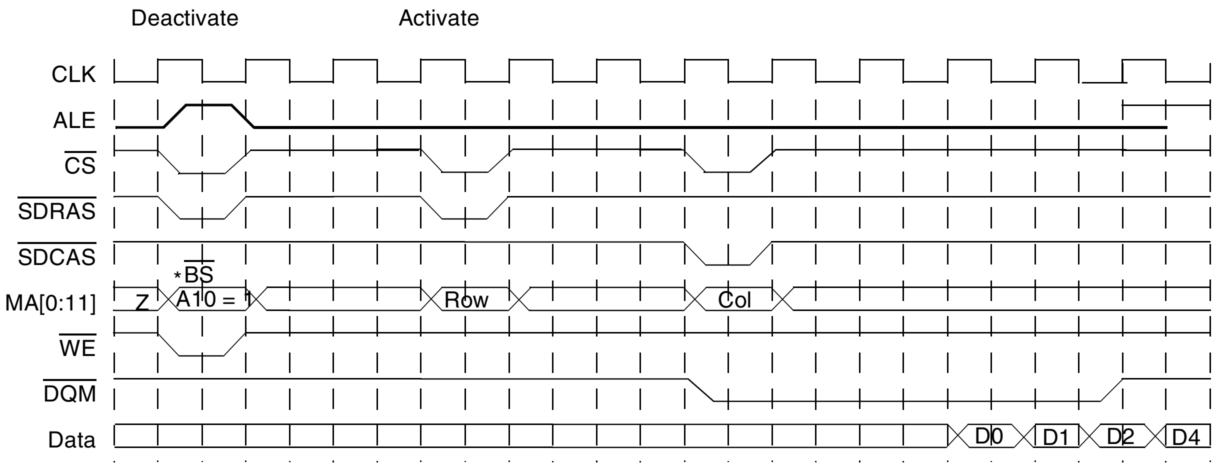


Figure 11-28. SDRAM Two-Beat Burst Read, Page Closed, CL = 3



* BS—Bank select according to SDRAM organization. A10 = 1 means all banks are precharged.

CAS Latency = 3

Figure 11-29. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

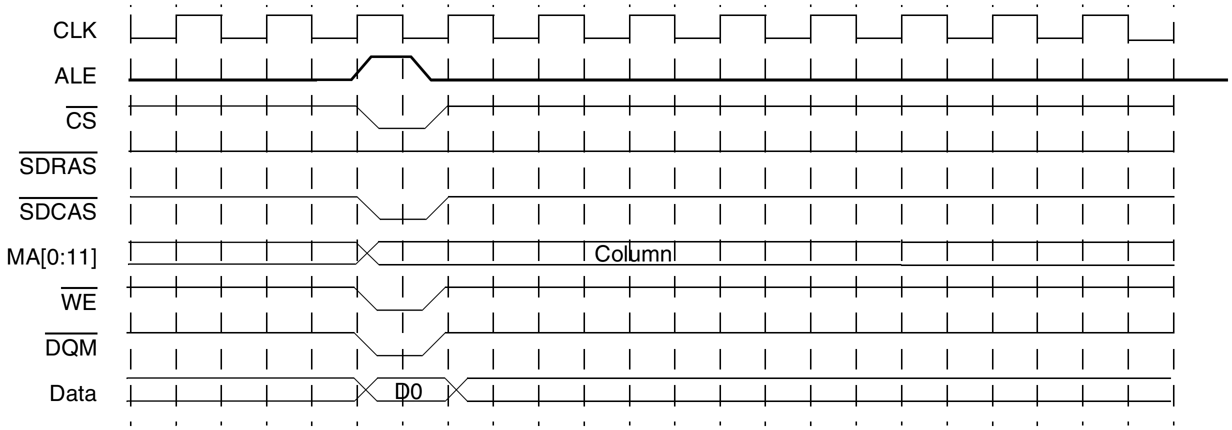


Figure 11-30. SDRAM Single-Beat Write, Page Hit

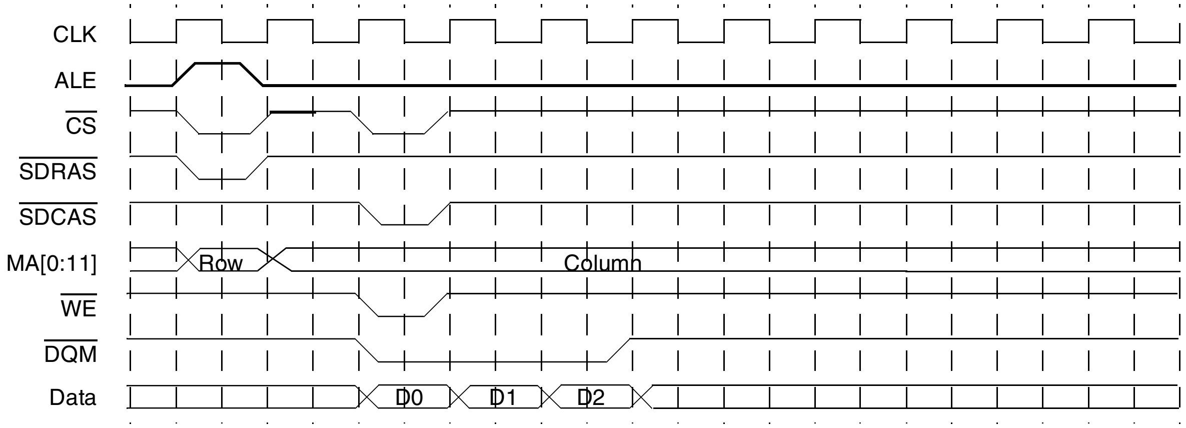
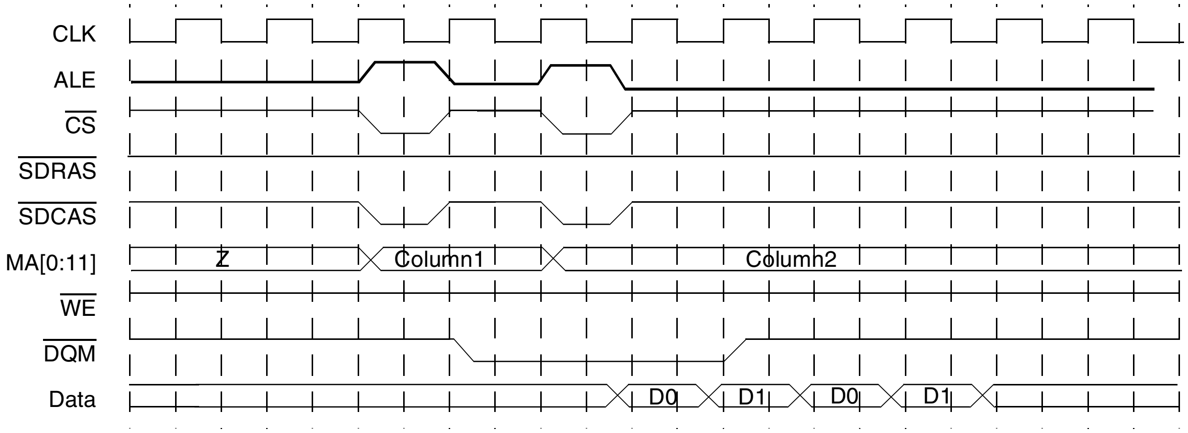


Figure 11-31. SDRAM Three-Beat Burst Write, Page Closed



DQM Latency (Affects Negation Only) = 2

Figure 11-32. SDRAM Read-after-Read Pipeline, Page Hit, CL = 3

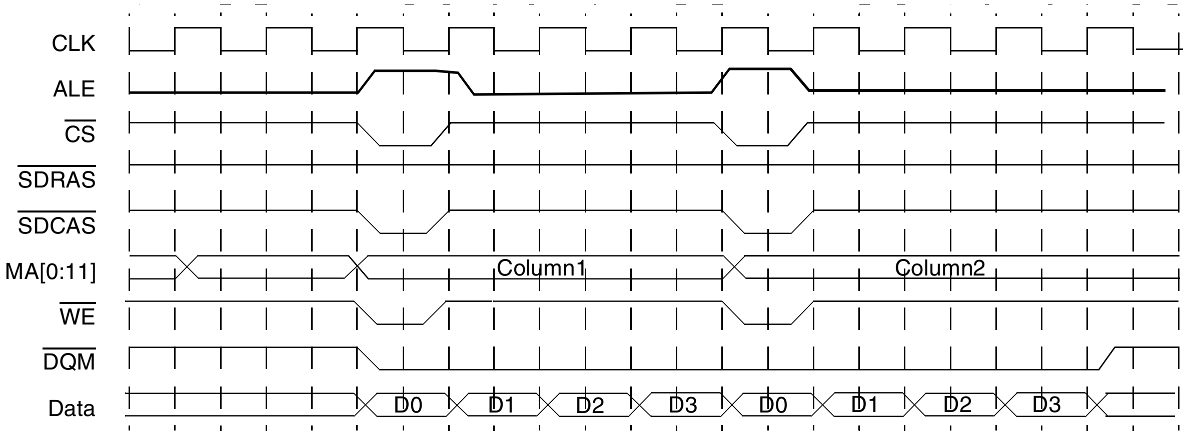


Figure 11-33. SDRAM Write-after-Write Pipelined, Page Hit

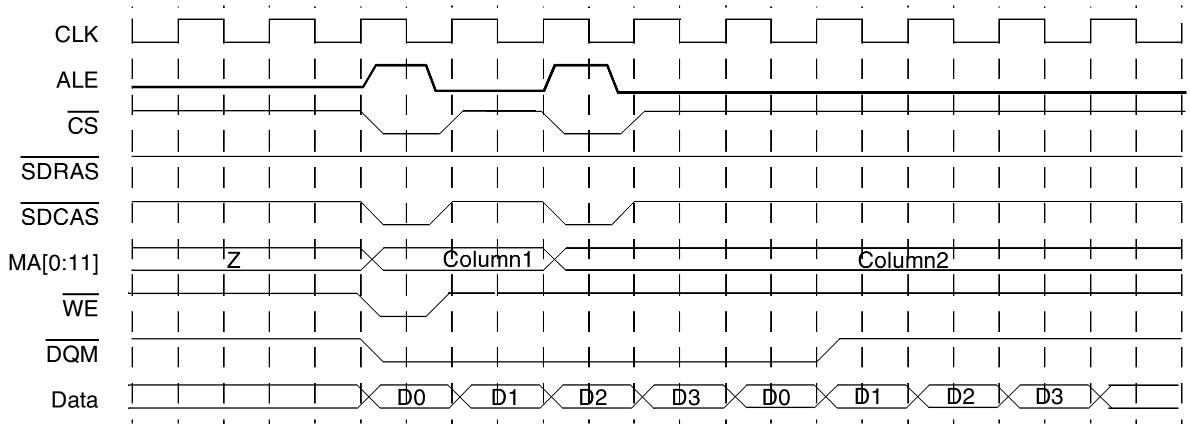


Figure 11-34. SDRAM Read-after-Write Pipelined, Page Hit

11.4.8 SDRAM Read/Write Transactions

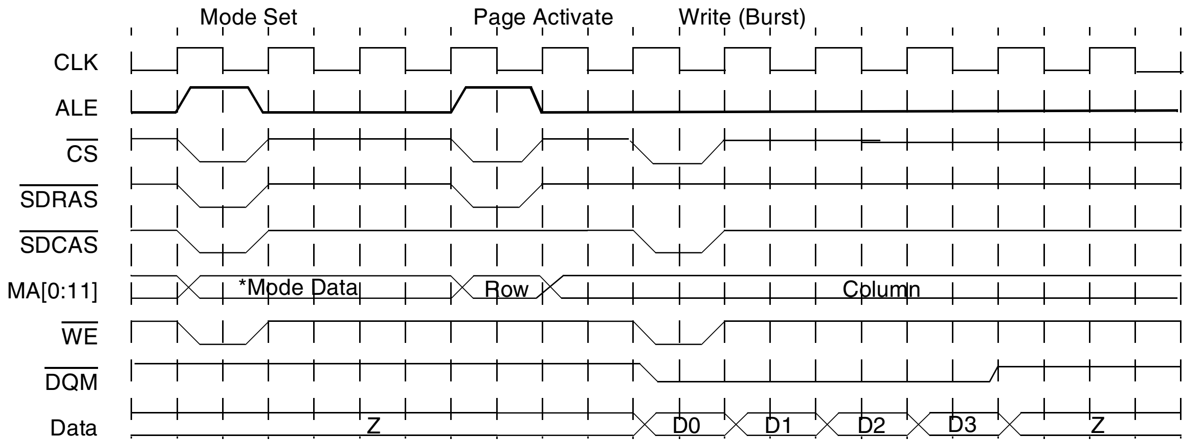
The SDRAM interface supports the following read/write transactions:

- Single-beat reads/writes up to double-word size
- Bursts of two, three, or four double words

SDRAM devices perform bursts for each transaction, the burst length depends on the port size. For 64-bit port size, it is a burst of 4. For 32-bit port size, it is a burst of 8. For reads that require less than the full burst length, extraneous data in the burst is ignored. For writes that require less than the full burst length, the MPC8272 protects non-targeted addresses by driving $\overline{\text{DQMn}}$ high on the irrelevant cycles of the burst. However, system performance is not compromised since, if a new transaction is pending, the MPC8272 begins executing it immediately, effectively terminating the burst early.

11.4.9 SDRAM MODE-SET Command Timing

The MPC8272 transfers mode register data (CAS latency, burst length, burst type) stored in P/LSDMR to the SDRAM array by issuing the MODE-SET command. Figure 11-35 shows timing for the MODE-SET command.



*The mode data is the address value during a mode-set cycle. It is driven by the memory controller, in single MPC8272 mode, according to P/LSDMR[CL] register. In 60x-compatible mode, software must drive the correct value on the address lines. Figure 11-36. shows the actual value.

Figure 11-35. SDRAM MODE-SET Command Timing

Figure 11-36 shows mode data bit settings.

Bit Number	0	1	2	3	4	5	6	7	8	9	10	11	lsb
							CL		0		BL		

Burst Length:
4(010) for 16- and 64-bit port sizes
8(011) for 8- and 32-bit port sizes

Latency Mode—can be 1(001), 2(010), or 3(011).

Figure 11-36. Mode Data Bit Settings

11.4.10 SDRAM Refresh

The memory controller supplies auto (CBR) refreshes to SDRAM according to the interval specified in PSRT or LSRT. This represents the time period required between refreshes. The value of P/LSRT depends on the specific SDRAM devices used and the operating frequency of the MPC8272's bus. This value should allow for a potential collision between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires, this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

NOTE

There are two SDRAM refresh timers, one for 60x SDRAM machines.

11.4.11 SDRAM Refresh Timing

The memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

Once a refresh request is granted the memory controller begins issuing auto-refresh command to each device associated with the refresh timer, in one-clock intervals. After the last REFRESH command is issued, the memory controller waits for the number of clocks written in the SDRAM machine’s mode register (RFRC in P/LSDMR). The timing is shown in Figure 11-37.

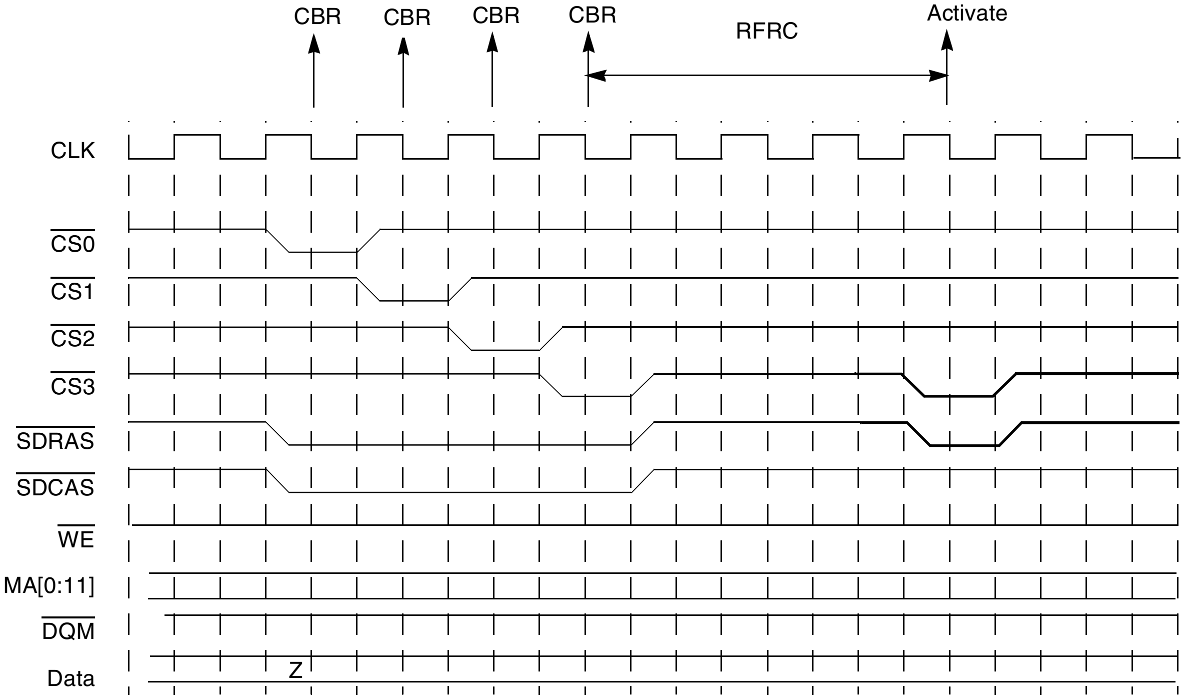


Figure 11-37. SDRAM Bank-Staggered CBR Refresh Timing

11.4.12 SDRAM Configuration Examples

The following sections provide SDRAM configuration examples for page- and bank-based interleaving.

11.4.12.1 SDRAM Configuration Example (Page-Based Interleaving)

Consider the following SDRAM organization:

- 64-bit port size, organized as eight 64-Mbit devices, each organized as 8M x 8bits.
- Each device has 4 internal banks, 12 rows, and 9 columns

For page-based interleaving, the address bus should be partitioned as shown in Table 11-19.

Table 11-19. 60x Address Bus Partition

A[0:5]	A[6:17]	A[18:19]	A[20:28]	A[29:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- PSDMR[PBI] = 1—Page-based interleaving
- ORx[BPD] = 01—Four internal banks
- ORx[ROWST] = 0110—Row starts at A[6]
- ORx[NUMR] = 011—Twelve row lines

Now, from the SDRAM device point of view, during an ACTIVATE command, its address port should look like Table 11-20.

Table 11-20. SDRAM Device Address Port during ACTIVATE Command

A[0:14]	A[15:16]	A[17:28]	A[29:31]
—	Internal bank select (A[18:19])	Row (A[6:17])	n.c.

Table 11-17 indicates that to multiplex A[6:17] over A[17:28], PSDMR[SDAM] must be 011 and, because the internal bank selects are multiplexed over A[15:16], PSDMR[BSMA] must be 010 (only the lower two bank select lines are used).

When using bank-based interleaving, the internal bank-select signals that are multiplexed over the address lines should be adjacent to the row address during the ACTIVATE command (refer to Table 11-20). So, the value of PSDMR[BSMA] is selected according to the combination of PSDMR[SDAM], ORx[ROWST], and ORx[NUMR]. Otherwise, the output of the BNKSEL pins could be incorrect even if the device is connected to the BNKSEL pins. To ensure proper connection, note that BNKSEL0 is msb and BNKSEL2 is lsb as stated in Table 6-1.

When using page-based interleaving, the internal bank-select signals that are multiplexed over the address lines are determined only by PSDMR[BSMA] during the ACTIVATE command. The output of the BNKSEL pins are not affected by the PSDMR[BSMA] value.

During a READ/WRITE command, the address port should look like Table 11-21.

Table 11-21. SDRAM Device Address Port during READ/WRITE Command

A[0:14]	A[15:16]	A[17]	A[18]	A[19]	A[20:28]	A[29:31]
—	Internal bank select	Do not care	AP	Do not care	Column	n.c.

Because AP alternates with A[7] of the row lines, set PSDMR[SDA10] = 011. This outputs A[7] on the SDA10 line during the ACTIVATE command and AP during READ/WRITE and CBR commands.

Table 11-22 shows the register configuration. Not shown are PSRT and MPTPR, which should be programmed according to the device refresh requirements:

Table 11-22. Register Settings (Page-Based Interleaving)

Register	Settings	
BR*x*	[BA] = Base address [PS00] = 64-bit port size [DECC] = 00 [WP] = 0 [MS010] = SDRAM-60x bus	[EMEMC] =0 [ATOM] = 00 [DR] = 0 [V] = 1
OR*x*	[AM] =1111_1100_0000 [LSDAM] =00000 [BPD] = 01 [ROWST] = 0110	[NUMR] =011 [PMSEL] =0 [IBID] = 0
PSDMR	[PBI] =1 [RFEN] =1 [OP] = 000 [SDAM] =011 [BSMA] =010 [SDA10] =011 [RFRC] = From device data sheet [PRETOACT] =From device data sheet	[ACTTOROW] = From device data sheet [BL] =0 [LDOTOPRE] = From device data sheet [WRC] = From device data sheet [EAMUX] =0 [BUFCMD] =0 [CL] = From device data sheet

11.4.13 SDRAM Configuration Example (Bank-Based Interleaving)

Consider the following SDRAM organization:

- Eight 64-Mbit devices, each organized as 8M x 8 bits
- Each device has four internal banks, 12 rows, and 9 columns

For bank-based interleaving, this means that the address bus should be partitioned as shown in Table 11-23.

Table 11-23. 60x Address Bus Partition

A[0:5]	A[6:7]	A[8:19]	A[20:28]	A[29:31]
msb of start address	Internal bank select	Row	Column	lsb

The following parameters can be extracted:

- PSDMR[PBI] = 0
- ORx[BPD] = 01—4 internal banks
- ORx[ROWST] = 0100—row starts at A[8]
- ORx[NUMR] = 011—there are 12 row lines

Now, from the SDRAM device point of view, during an ACTIVATE command, its address port should look like Table 11-24.

Table 11-24. SDRAM Device Address Port during ACTIVATE Command

A[0:14]	A[15:16]	A[17:28]	A[29:31]
—	Internal bank select (A[6:7])	Row (A[8:19])	n.c.

Table 11-17 indicates that in order to multiplex A[6:19] over A[15:28] PSDMR[SDAM] must be 001 and, because the internal bank selects are multiplexed over A[15:16] PSDMR[BSMA] must be 010 (only the lower two bank select lines are used).

During a READ/WRITE command, the address port should look like Table 11-25.

Table 11-25. SDRAM Device Address Port during READ/WRITE Command

A[0:14]	A[15:16]	A[17]	A[18]	A[19]	A[20:28]	A[29:31]
—	Internal bank select	Do not care	AP	Do not care	Column	n.c.

Because AP alternates with A[9] of the row lines, set PSDMR[SDA10] = 011. This outputs A[9] on the SDA10 line during the ACTIVATE command and AP during READ/WRITE and CBR commands.

Table 11-26 shows the register configuration. Not shown are PSRT and MPTPR, which should be programmed according to the device refresh requirements.

Table 11-26. Register Settings (Bank-Based Interleaving)

Register	Settings	
BR*x*	[BA] = Base address [PS] = 00 (64-bit port size0 [DECC] = 00 [WP] = 0 [MS]010 = SDRAM-60x bus	[EMEMC] =0 [ATOM] = 00 [DR] = 0 [V] = 1
OR*x*	[SDAM] = 111_1100_0000 [LSDAM] = 00000 [BPD] = 01 [ROWST] = 010	[NUMR] = 011 [PMSEL] =0 [IBID] =0
PSDMR	[PBI] = 0 [RFEN] =1 [OP] = 000 [SDAM] = 001 [BSMA] = 010 [SDA] = 10011 [RFRC] From device data sheet [PRETOACT] From device data sheet	[ACTTOROW]From device data sheet [BL] = 0 [LDOTOPRE]From device data sheet [WRC] From device data sheet [EAMUX] =0 [BUFCMD] =0 [CL] From device data sheet

11.5 General-Purpose Chip-Select Machine (GPCM)

Users familiar with the MPC8xx memory controller should read Section 11.5.4, “Differences between MPC8xx’s GPCM and MPC82xx’s GPCM,” first.

The GPCM allows a glueless and flexible interface between the MPC8272, SRAM, EEPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BRx and ORx.

Although GPCM does not support bursting, the internal logic will split a burst into individual beats that the GPCM can support. Therefore, the flash can be cached. Note that if the MPC8272 is in 60x-bus compatible mode, BADDR[27:31] should be used instead of A[27:31].

Table 11-27 lists the GPCM interface signals on the 60x.

Table 11-27. GPCM Interfaces Signals

60x Bus	Comments
$\overline{\text{CS}}$[0:7]	Device select
$\overline{\text{PWE}}$[0:7]	Write enables for write cycles
$\overline{\text{POE}}$	Output enable for read cycles

GPCM-controlled devices can use $\overline{\text{BCTLx}}$ as read/write indicators. The $\overline{\text{BCTLx}}$ signals appears as R/$\overline{\text{W}}$ in the timing diagrams. See Section 11.2.4, "Data Buffer Controls (BCTLx)."

Additional control is available in 60x-compatible mode (60x bus only)—ALE–external address latch enable

In this section, the output-enable and write-enable signals are generically labeled $\overline{\text{OE}}$ and $\overline{\text{WE}}$. When using the 60x bus they refer to $\overline{\text{POE}}$ and $\overline{\text{PWE}}$.

Figure 11-38 shows a simple connection between a 32-bit port size SRAM device and the MPC8272.

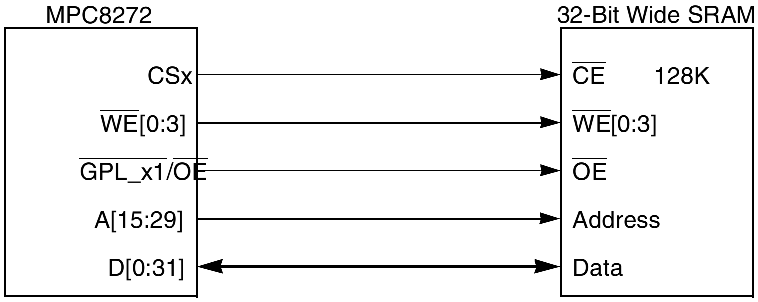


Figure 11-38. GPCM-to-SRAM Configuration

11.5.1 Timing Configuration

If BRx[MS] selects the GPCM, the attributes for the memory cycle are taken from ORx. These attributes include the CSNT, ACS[0–1], SCY[0–3], TRLX, EHTR, and SETA fields. Table 11-28 shows signal behavior and system response.

Table 11-28. GPCM Strobe Signal Behavior

Option Register Attributes				Signal Behavior			
TRLX	Access	ACS	CSNT	Address to $\overline{\text{CS}}$ Asserted	$\overline{\text{CS}}$ Negated to Address Change	$\overline{\text{WE}}$ Negated to Address/Data Invalid	Total Cycles
0	Read	00	x	0	0	x	2+SCY[1]
0	Read	10	x	1/4 × Clock	0	x	2+SCY
0	Read	11	x	1/2 × Clock	0	x	2+SCY
0	Write	00	0	0	0	0	2+SCY
0	Write	10	0	1/4 × Clock	0	0	2+SCY

Table 11-28. GPCM Strobe Signal Behavior (continued)

Option Register Attributes				Signal Behavior			
TRLX	Access	ACS	CSNT	Address to $\overline{\text{CS}}$ Asserted	$\overline{\text{CS}}$ Negated to Address Change	$\overline{\text{WE}}$ Negated to Address/Data Invalid	Total Cycles
0	Write	11	0	1/2 × Clock	0	0	2+SCY
0	Write	00	1	0	0	–1/4 × Clock	2+SCY
0	Write	10	1	1/4 × Clock	–1/4 × Clock	–1/4 × Clock	2+SCY
0	Write	11	1	1/2 × Clock	–1/4 × Clock	–1/4 × Clock	2+SCY
1	Read	00	x	0	0	x	2+2 × SCY
1	Read	10	x	(1+1/4) × Clock	0	x	3+2 × SCY
1	Read	11	x	(1+1/2) × Clock	0	x	3+2 × SCY
1	Write	00	0	0	0	0	2+2 × SCY
1	Write	10	0	(1+1/4) × Clock	0	0	3+2 × SCY
1	Write	11	0	(1+1/2) × Clock	0	0	3+2 × SCY
1	Write	00	1	0	0	–1–1/4 × Clock	3+2 × SCY
1	Write	10	1	(1+1/4) × Clock	–1–1/4 × Clock	–1–1/4 × Clock	4+2 × SCY
1	Write	11	1	(1+1/2) × Clock	–1–1/4 × Clock	–1–1/4 × Clock	4+2 × SCY

[1] SCY is the number of wait cycles from the option register.

11.5.1.1 Chip-Select Assertion Timing

From 0 to 30 wait states can be programmed for $\overline{\text{PSDVAL}}$ generation. Byte-write enable signals ($\overline{\text{WE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{OE}}$) is provided to eliminate external glue logic. The memory banks selected to work with the GPCM have unique features. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select prior to the system being fully configured. The banks selected to work with the GPCM support an option to output the $\overline{\text{CS}}$ line at different timings with respect to the external address bus. $\overline{\text{CS}}$ can be output in any of three configurations:

- Simultaneous with the external address
- One quarter of a clock cycle later
- One half of a clock cycle later

Note the GPCM does not negate $\overline{\text{CS}}$ in back-to-back reads to the same device when in single MPC8272 bus mode or in 60x-compatible bus mode with extended transfers enabled. When in strict 60x bus mode, however, the GPCM does negate $\overline{\text{CS}}$ in back-to-back reads. See Section 4.3.2.1, “Bus Configuration Register (BCR).”

Figure 11-39 shows a basic connection between the MPC8272 and an external peripheral device. Here, $\overline{\text{CS}}$ (the strobe output for the memory access) is connected directly to $\overline{\text{CE}}$ of the memory device and BCTL0 is connected to the respective R/$\overline{\text{W}}$ in the peripheral device.

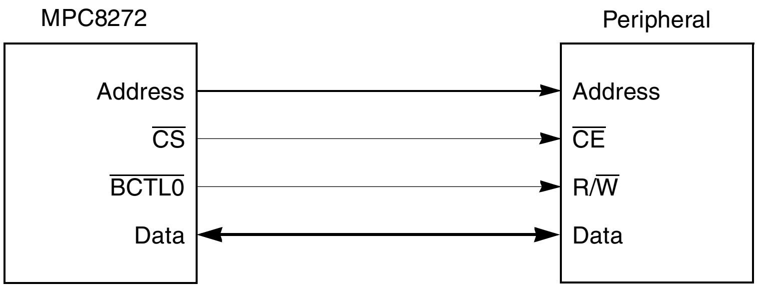


Figure 11-39. GPCM Peripheral Device Interface

Figure 11-40 shows $\overline{\text{CS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure ORx[ACS] to specify $\overline{\text{CS}}$ to meet this requirement.

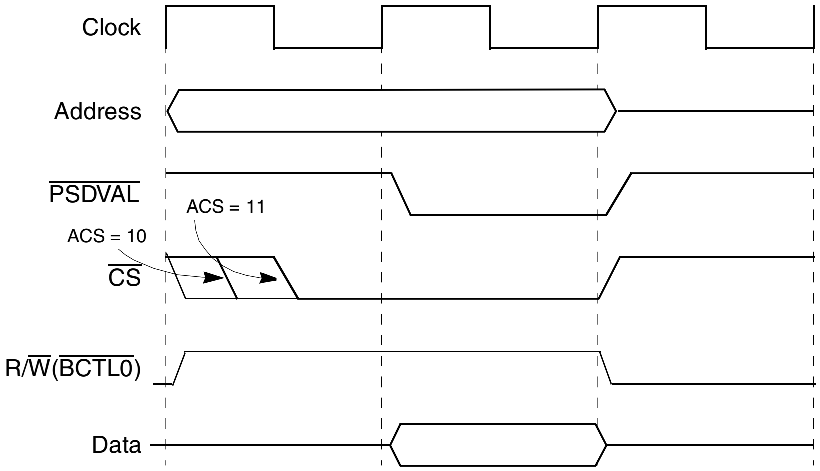

Clock
Address
$\overline{\text{PSDVAL}}$
ACS = 11
ACS = 10
$\overline{\text{CS}}$
R/$\overline{\text{W}}$($\overline{\text{BCTL0}}$)
Data

Figure 11-40. GPCM Peripheral Device Basic Timing (ACS = 1x and TRLX = 0)

11.5.1.2 Chip-Select and Write Enable Deassertion Timing

Figure 11-41 shows a basic connection between the MPC8272 and a static memory device. Here, $\overline{\text{CS}}$ is connected directly to $\overline{\text{CE}}$ of the memory device. The $\overline{\text{WE}}$ signals are connected to the respective $\overline{\text{W}}$ signal in the memory device where each $\overline{\text{WE}}$ corresponds to a different data byte.

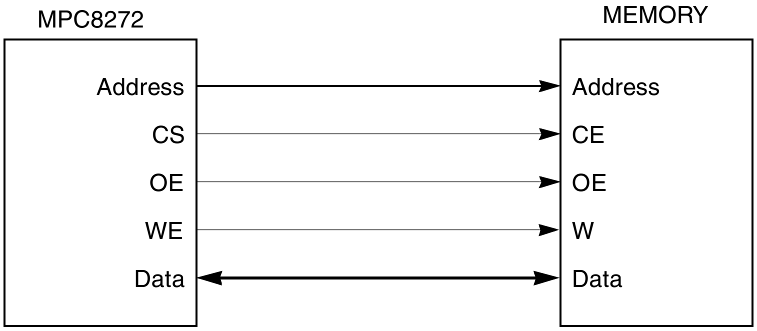


Figure 11-41. GPCM Memory Device Interface

As Figure 11-43 shows, the timing for $\overline{\text{CS}}$ is the same as for the address lines. The strobes for the transaction are supplied by $\overline{\text{OE}}$ or $\overline{\text{WE}}$, depending on the transaction direction (read or write). ORx[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. For example, when ACS = 00 and CSNT = 1, $\overline{\text{WE}}$ is negated one quarter of a clock earlier, as shown in Figure 11-42.

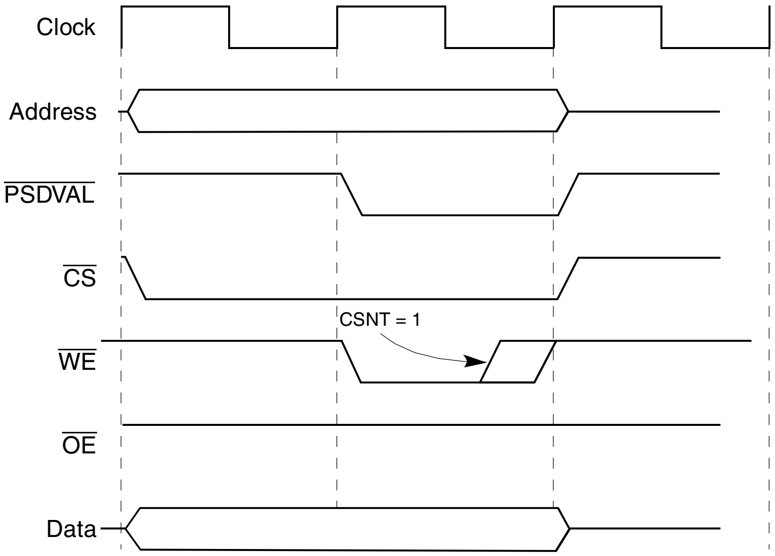


Figure 11-42. GPCM Memory Device Basic Timing (ACS = 00, CSNT = 1, TRLX = 0)

When ACS ≠ 00 and CSNT = 1, $\overline{\text{WE}}$ and $\overline{\text{CS}}$ are negated one quarter of a clock earlier, as shown in Figure 11-43.

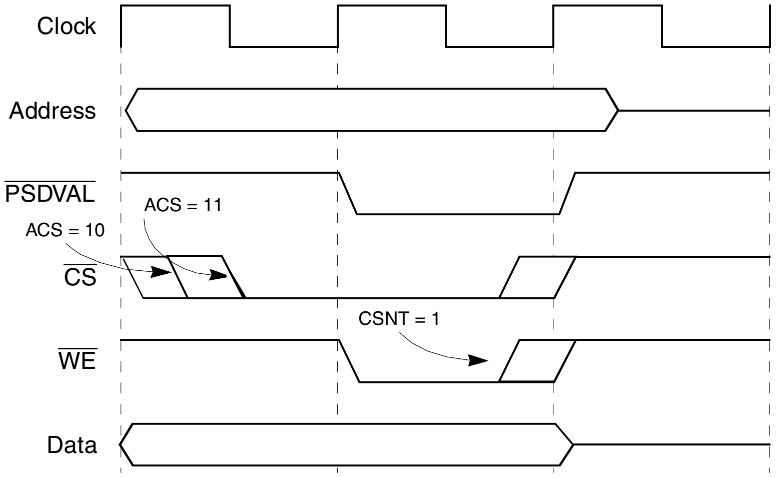


Figure 11-43. GPCM Memory Device Basic Timing (ACS ≠ 00, CSNT = 1, TRLX = 0)

11.5.1.3 Relaxed Timing

ORx[TRLX] is provided for memory systems that require more relaxed timing between signals. When TRLX = 1 and ACS ≠ 00, an additional cycle between the address and strobes is inserted by the MPC8272 memory controller. See Figure 11-44 and Figure 11-45.

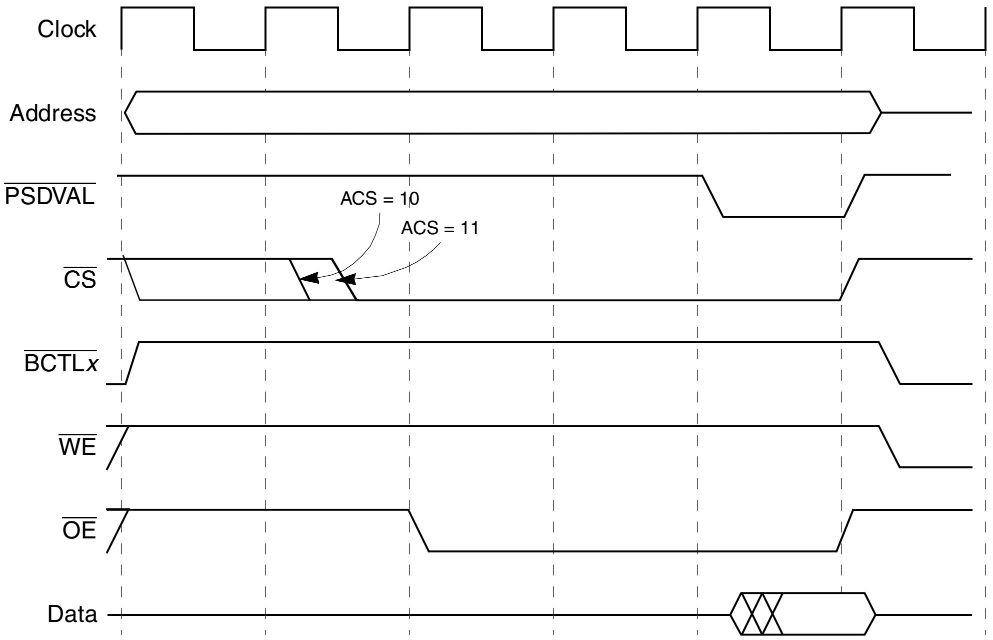


Figure 11-44. GPCM Relaxed Timing Read (ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1)

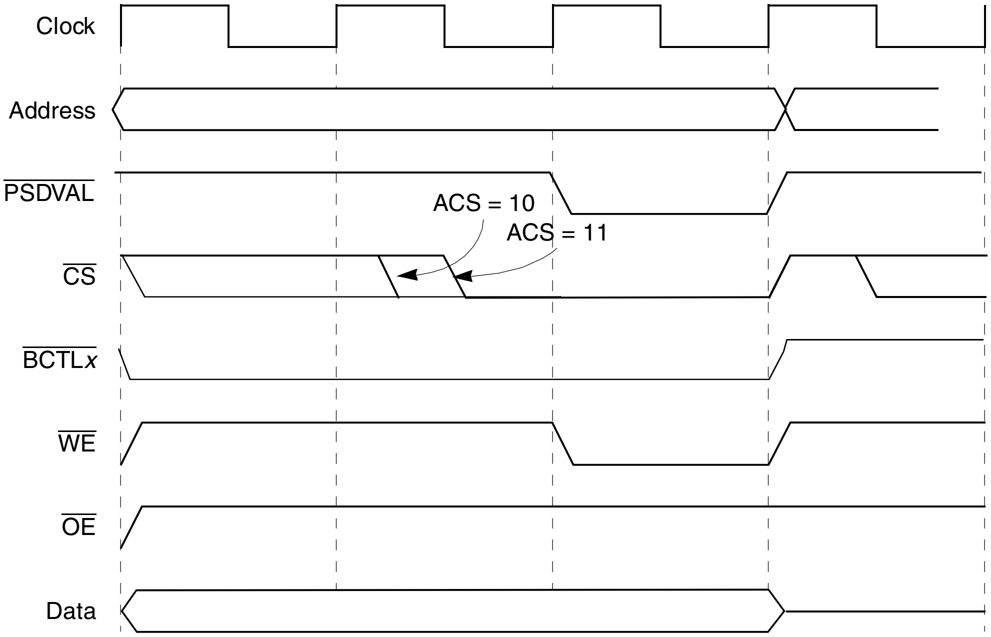


Figure 11-45. GPCM Relaxed-Timing Write (ACS = 1x, SCY = 0, CSNT = 0,TRLX = 1)

When TRLX and CSNT are set in a write-memory access, the strobe lines, $\overline{\text{WE}}$[0:7] are negated one clock earlier than in the normal case. If ACS ≠ 0, $\overline{\text{CS}}$ is also negated one clock earlier, as shown in Figure 11-46 and Figure 11-47. When a bank is selected to operate with external transfer acknowledge (SETA and TRLX = 1), the memory controller does not support external devices that provide $\overline{\text{PSDVAL}}$ to complete the transfer with zero wait states. The minimum access duration in this case is 3 clock cycles.

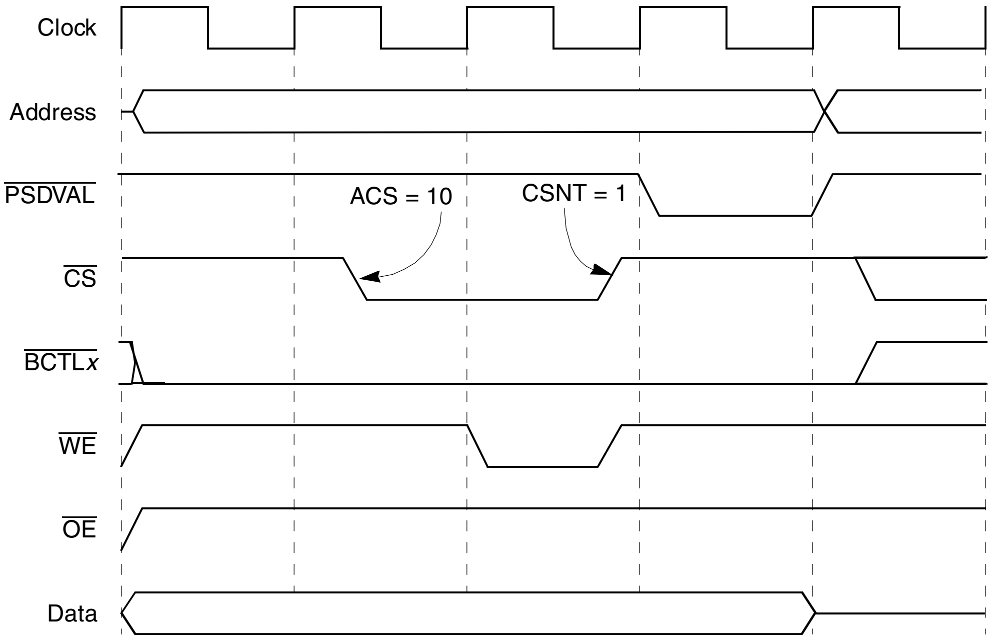


Figure 11-46. GPCM Relaxed-Timing Write (ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)

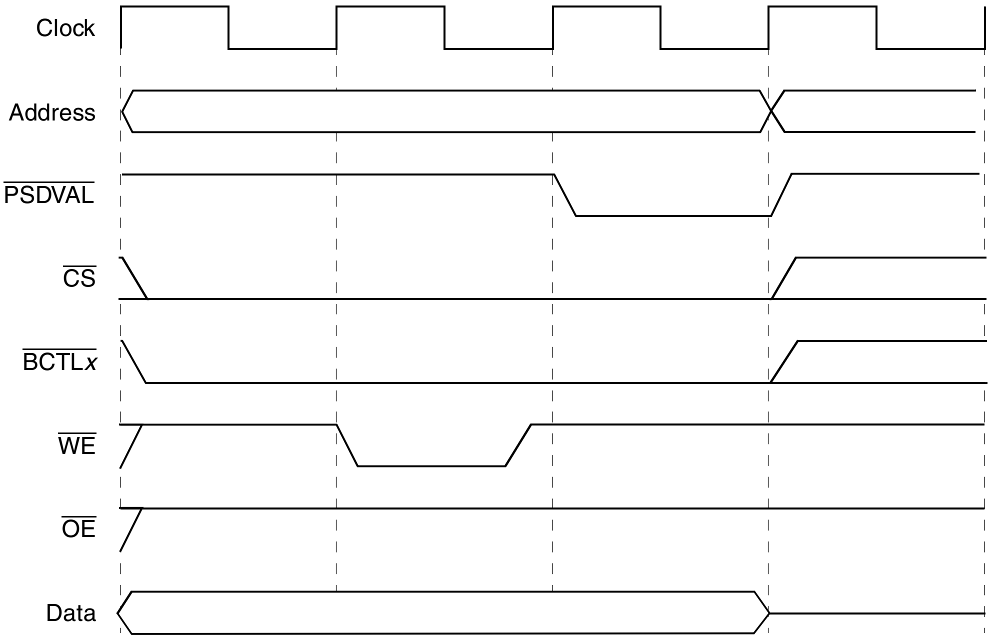


Figure 11-47. GPCM Relaxed-Timing Write (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1)

11.5.1.4 Output Enable ($\overline{\text{OE}}$) Timing

The timing of the $\overline{\text{OE}}$ is affected only by TRLX. It always asserts and negates on the rising edge of the external bus clock. $\overline{\text{OE}}$ always asserts on the rising clock edge after $\overline{\text{CS}}$ is asserted, and therefore its assertion can be delayed (along with the assertion of $\overline{\text{CS}}$) by programming TRLX = 1. $\overline{\text{OE}}$ deasserts on the rising clock edge coinciding with or immediately after $\overline{\text{CS}}$ deassertion.

11.5.1.5 Programmable Wait State Configuration

The GPCM supports internal $\overline{\text{PSDVAL}}$ generation. It allows fast accesses to external memory through an internal bus master or a maximum 17-clock access by programming ORx[SCY]. The internal $\overline{\text{PSDVAL}}$ generation mode is enabled if ORx[SETA] = 0. If $\overline{\text{GTA}}$ is asserted externally at least two clock cycles before the wait state counter has expired, the current memory cycle is terminated. When TRLX = 1, the number of wait states inserted by the memory controller is defined by 2 x SCY or a maximum of 30 wait states.

11.5.1.6 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 combination of ORx[29:30] (TRLX and EHTR). Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified by Table 11-29.

Table 11-29. TRLX and EHTR Combinations

ORx[TRLX]	ORx[EHTR]	Number of Hold Time Clock Cycles
0	0	0
0	1	1
1	0	4
1	1	8

See Figure 11-48 through Figure 11-51 for timing examples.

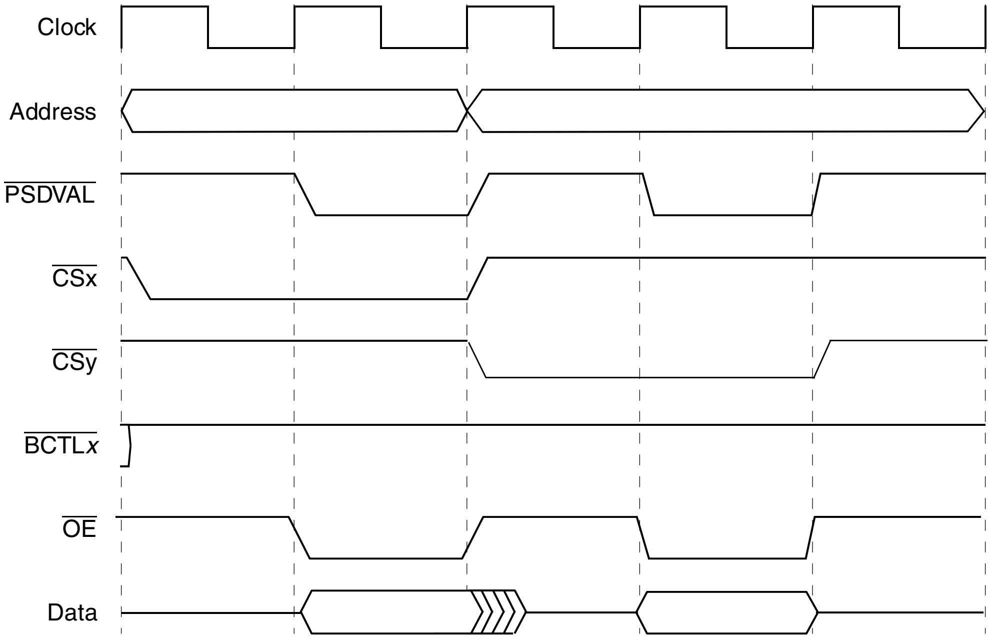


Figure 11-48. GPCM Read Followed by Read (OR*x*[29:30] = 00, Fastest Timing)

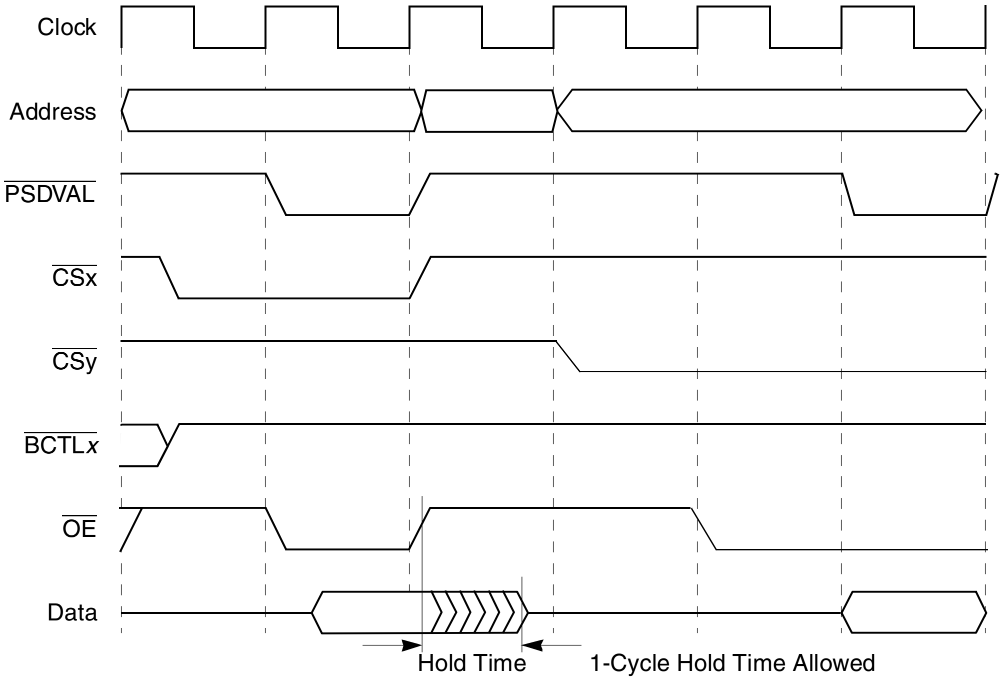


Figure 11-49. GPCM Read Followed by Read (OR*x*[29:30] = 01)

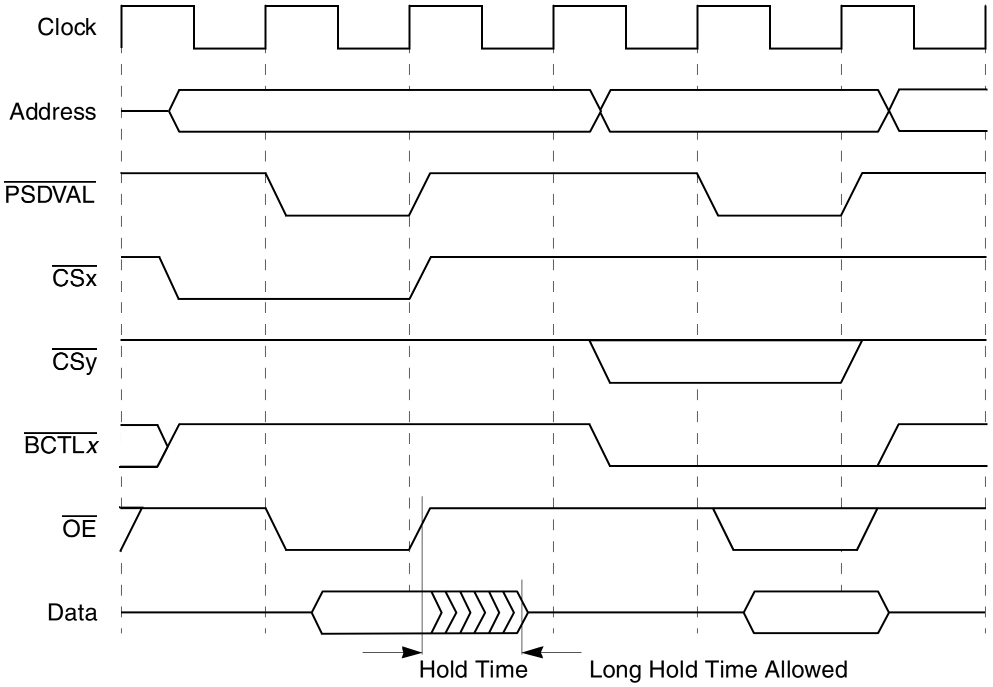


Figure 11-50. GPCM Read Followed by Write (OR*x*[29:30] = 01)

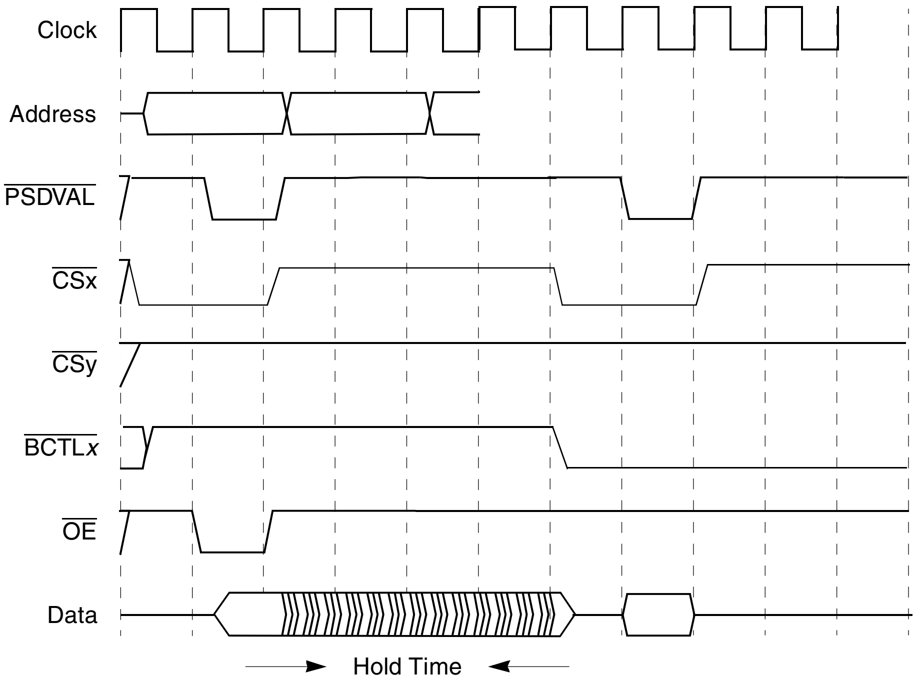


Figure 11-51. GPCM Read Followed by Write (OR*x*[29:30] = 10)

11.5.2 External Access Termination

External access termination is supported by the GPCM using GTA, which is synchronized and sampled internally by the MPC8272. If, during a GPCM data phase (second cycle or later), the sampled signal is asserted, it is converted to PSDVAL, which terminates the current GPCM access. GTA should be asserted for one cycle. Note that because GTA is synchronized, bus termination may occur three cycles after GTA assertion, so in case of read cycle, the device still must output data as long as OE is asserted. The user selects whether PSDVAL is generated internally or externally (by means of GTA assertion) by resetting/setting ORx[SETA].

Figure 11-52 shows how a GPCM access is terminated by GTA assertion. Asserting GTA terminates an access even if ORx[SETA] = 0 (internal PSDVAL generation).

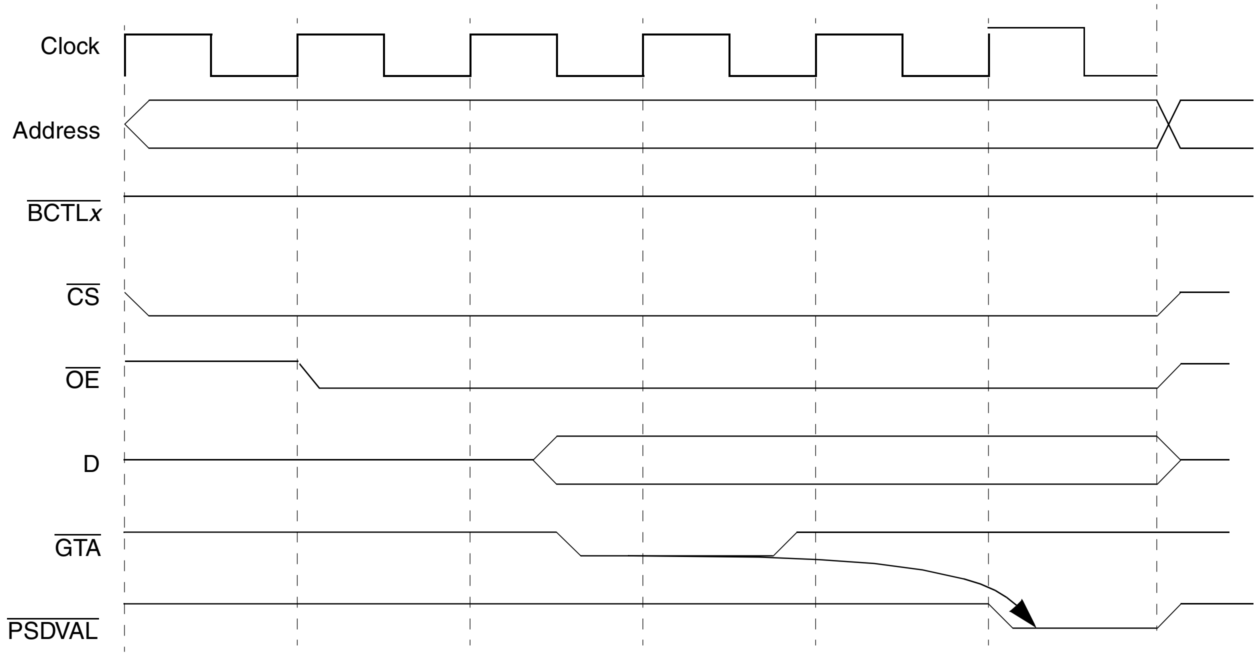


Figure 11-52. External Termination of GPCM Access

11.5.3 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. The CS0 signal is the boot chip-select output; its operation differs from the other external chip-select outputs on system reset. When the MPC8272 internal core begins accessing memory at system reset, CS0 is asserted for every address in the boot address range, unless an internal register is accessed. The address range is configured during reset.

The boot chip-select also provides a programmable port size during system reset by using the configuration mechanism described in Section 5.4, "Reset Configuration." The boot chip-select does not provide write protection. CS0 operates this way until the first write to OR0 and it can be used as any other chip-select register once the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only on hardware reset. Table 11-30 describes the initial values of the boot bank in the memory controller.

Table 11-30. . Boot Bank Field Values after Reset

Register	Setting
BR0	BA From hard reset configuration word. See Section 5.4.1, "Hard Reset Configuration Word." PS From hard reset configuration word. See Section 5.4.1, "Hard Reset Configuration Word." DECC 0 WP 0 MS 000 EMEMC From hard reset configuration word. See Section 5.4.1, "Hard Reset Configuration Word." V 1

Table 11-30. . Boot Bank Field Values after Reset (continued)

Register	Setting
OR0	AM1111_1110_0000_0000_0 (32 Mbytes) BCTLD0 CSNT1 ACS11 SCY1111 SETA0 TRLX1 EHTR0

11.5.4 Differences between MPC8xx's GPCM and MPC82xx's GPCM

Users familiar with the MPC8xx GPCM should read this section first:

- External termination—In the MPC8xx the external termination connects to the external bus $\overline{\text{TA}}$ and so must be asserted in sync with the system clock. In the MPC8272, this signal is separated from the bus and named $\overline{\text{GTA}}$. The signal is synchronized internally and sampled. The sampled signal is used to generate $\overline{\text{TA}}$, which terminates the bus transaction.
- Extended hold time for reads can be up to 8 clock cycles (instead of 1 in the MPC8xx).

11.6 User-Programmable Machines (UPMs)

Users familiar with MPC8xx memory controller should first read Section 11.6.6, "Differences between MPC8xx UPM and MPC82xx UPM." Table 11-31 lists the UPM interface signals on the 60x.

Table 11-31. UPM Interfaces Signals

60x Bus	Comments
$\overline{\text{CS}}$[0:7]	Device select
$\overline{\text{PBS}}$[0:7]	Byte Select
PGPL_0	General-purpose line 0
PGPL_1	General-purpose line 1
PGPL_2	General-purpose line 2
PGPL_3	General-purpose line 3
PGPL_4/UPMWAIT	General-purpose line 4/UPM wait
PGPL_5	General-purpose line 5

Additional control is available in 60x-compatible mode (60x bus only) using the external address latch enable (ALE). However, ALE is not a UPM-controlled signal; it toggles with chip select signals.

Note that in this section, when a signal is named, the reference is to the 60x bus.

The three user-programmable machines (UPMs) are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal-memory RAM array that specifies the logical value driven on the external memory controller pins 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 11-53 shows the basic operation of each UPM. The following events initiate a UPM cycle:

- Any internal or external 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 transfer error or reset generates an exception request.

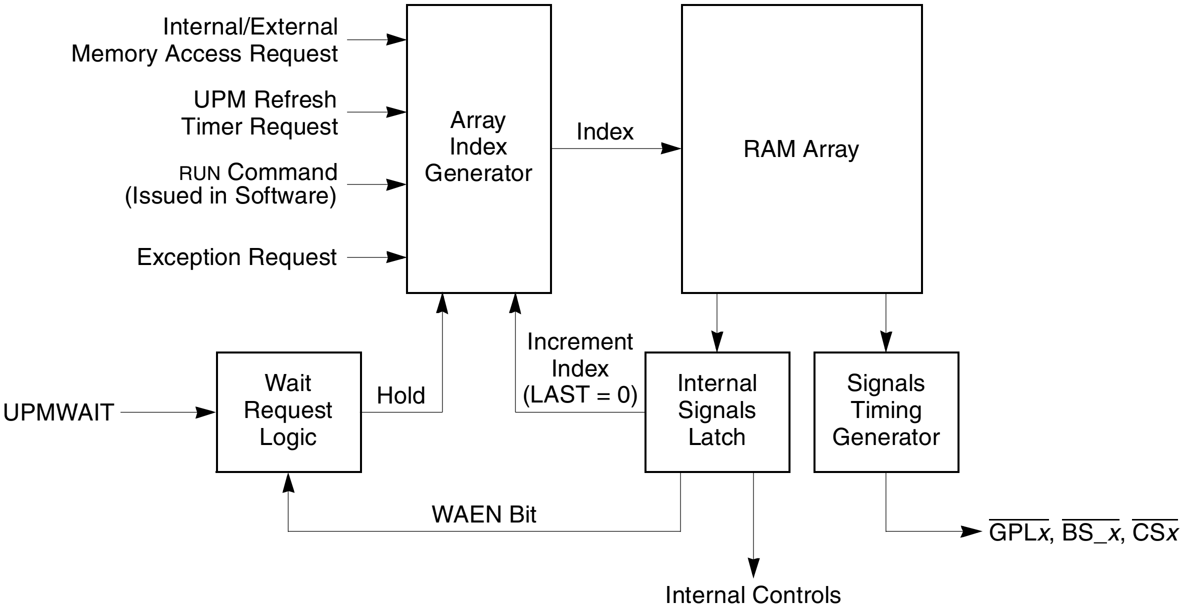


Figure 11-53. User-Programmable Machine Block Diagram

The RAM array contains 64 32-bit RAM words. 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 UPMWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

When a new access to external memory is requested by any device on the 60x bus, the addresses of the transfer are compared to each one of the valid banks defined in the memory controller. When an address match is found in one of the memory banks, BRx[MS] selects the UPM to handle this memory access. MxMR[BSEL] assigns the UPM to the 60x bus.

11.6.1 Requests

An internal or external device's request for a memory access initiates one of the following patterns (MxMR[OP] = 00):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

These patterns are described in Section 11.6.1.1, “Memory Access Requests.”

A UPM refresh timer request pattern initiates a refresh timer pattern (PTS), as described in Section 11.6.1.2, “UPM Refresh Timer Requests.”

An exception (caused by a soft reset or the assertion of $\overline{\text{TEA}}$) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

A special pattern in the RAM array is associated with each of these cycle type. Figure 11-54 shows the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (MxMR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

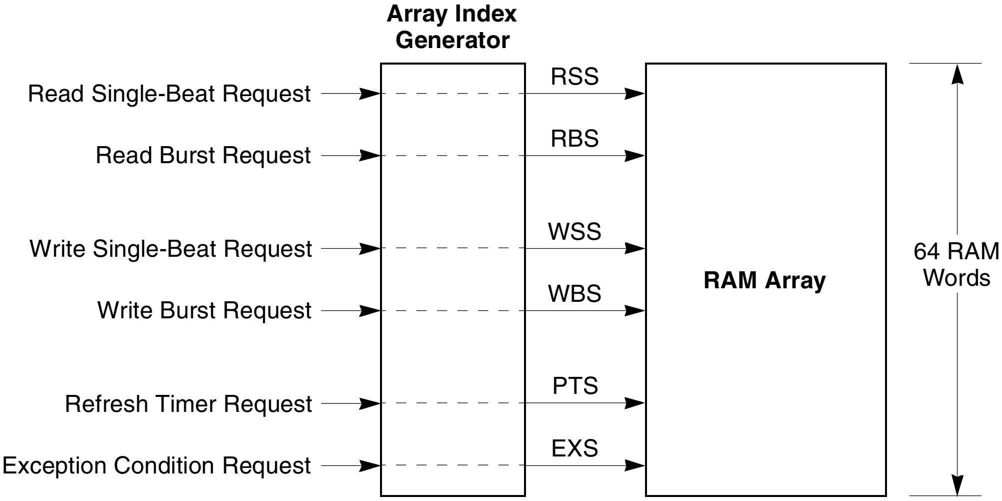


Figure 11-54. RAM Array Indexing

Table 11-32 show the start address of each pattern.

Table 11-32. 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 (PTS)	0x30
Exception condition (EXS)	0x3C

11.6.1.1 Memory Access Requests

When an internal device requests a new access to external memory, the address of transfer is compared to each valid bank defined in BRx. The value in BRx[MS] selects the UPM to handle the memory access. The user must ensure that the UPM is appropriately initialized before a request.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to double word. A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers four double words. For 64-bit accesses, the burst cycle starts with one transfer start but ends after four transfer acknowledges. A 32-bit device requires 8 data acknowledges; an 8-bit device requires 32. See Section 11.2.10, “Partial Data Valid Indication (PSDVAL).”

The MPC8272 defines two additional transfer sizes: bursts of two and three double words. These accesses are treated by the UPM as back-to-back, single-beat transfers.

11.6.1.2 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 11-55 shows the hardware associated with memory refresh timer request generation. PURT defines the period for the timers associated with UPMx on the 60x bus. See Section 11.3.7, “60x Bus-Assigned UPM Refresh Timer (PURT).”

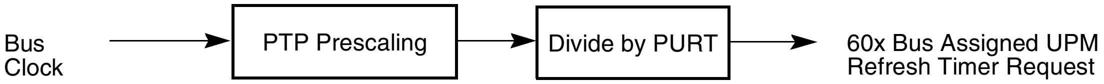


Figure 11-55. Memory Refresh Timer Request Block Diagram

All 60x bus refreshes are done using the refresh pattern of UPMA. This means that if refresh is required on the 60x bus, UPMA must be assigned to the 60x bus and MAMR[RFEN] must be set. It also means that only one refresh routine should be programmed for the 60x bus and be placed in UPMA, which serves as the 60x bus refresh executor. If refresh is not required on the 60x bus, UPMA can be assigned to any bus.

UPMB and UPMC can be assigned to the 60x bus if needed; there is no need to program its refresh routine because it will use the one in UPMA.

11.6.1.3 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 the 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 MxMR[OP] = 11 and accessing the UPMx memory region with a single-byte transaction.

Note that the pattern must contain exactly one assertion of $\overline{\text{PSDVAL}}$ (UTA bit in the RAM word, described in Table 11-33.), otherwise bus timeout may occur.

11.6.1.4 Exception Requests

When the MPC8272 under UPM control initiates an access to a memory device, the external device may assert $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$. The UPM provides a mechanism by which memory control signals can meet the timing requirements of the device without losing data. The mechanism is the exception pattern that defines how the UPM deasserts its signals in a controlled manner.

11.6.2 Programming the UPMs

The UPM is a microsequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program the UPMs:

1. Set up BRx and ORx.
2. Write patterns into the RAM array.
3. Program MPTPR and L/PURT if refresh is required.
4. Program the machine mode register (MxMR).

Patterns are written to the RAM array by setting MxMR[OP] = 01 and accessing the UPM with a single byte transaction. See Figure 11-11.

11.6.3 Clock Timing

Fields in the RAM word specify the value of the various external signals at each clock edge. The signal timing generator causes external signals to behave according to the timing specified in the current RAM word. Figure 11-56 and Figure 11-57 show the clock schemes of the UPMs in the memory controller for integer and non-integer clock ratios. The clock phases shown reflect timing windows during which generated signals can change state. If specified in the RAM, the value of the external signals can be changed after any of the positive edges of T[1:4], plus a circuit delay time as specified in the hardware specifications.

NOTE

> For integer clock ratios, the widths of T1/2/3/4 are equal, for a 1:2.5 clock ratio, T1 = 4/3 × T2 and T3 = 4/3 × T4, and for a 1:3.5 clock ratio, the ticks widths are T1 = 3/2 × T2 and T3 = 3/2 × T4.

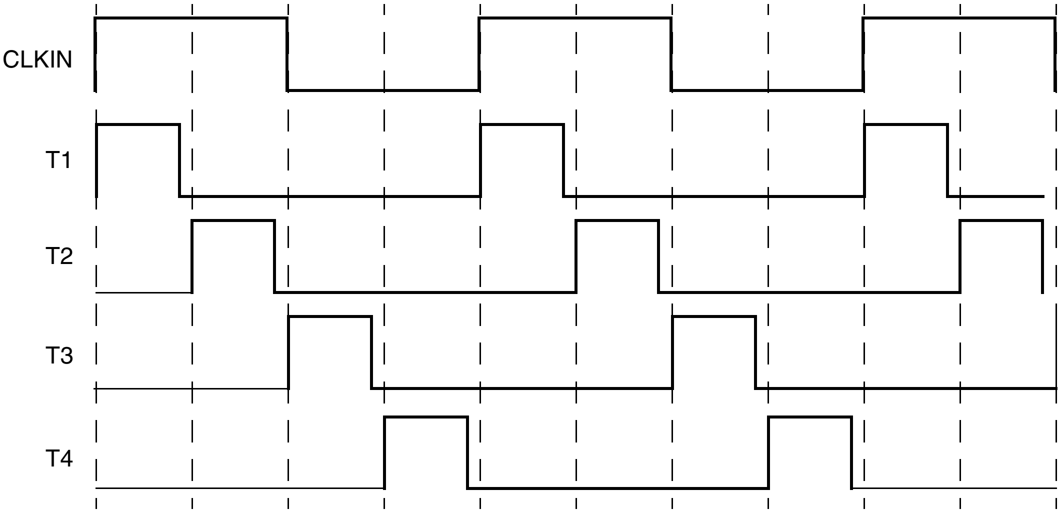


Figure 11-56. Memory Controller UPM Clock Scheme for Integer Clock Ratios

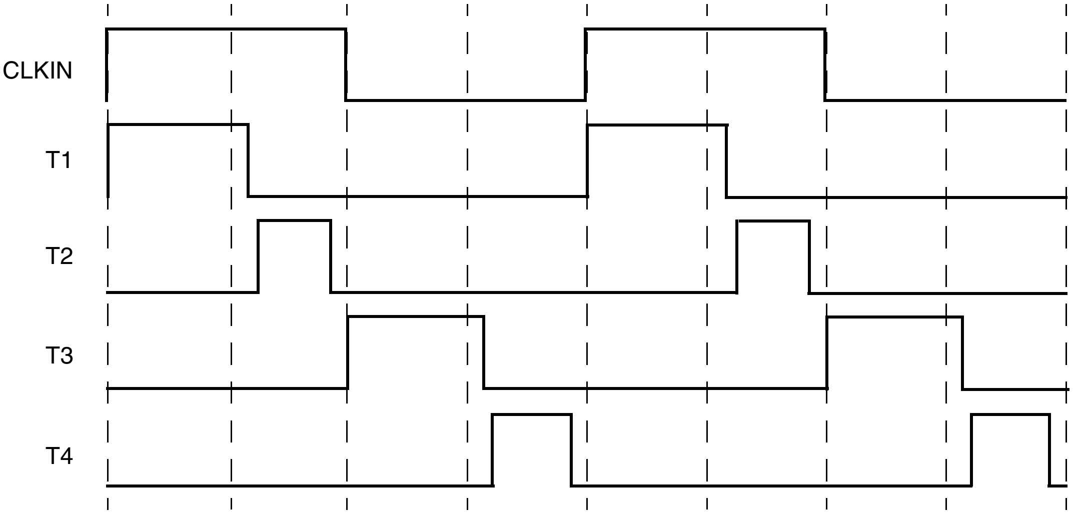


Figure 11-57. Memory Controller UPM Clock Scheme for Non-Integer (2.5:1/3.5:1) Clock Ratios

The state of the external signals may change (if specified in the RAM array) at any positive edge of T1, T2, T3, or T4 (there is a propagation delay specified in the hardware specifications). Note however that only the $\overline{\text{CS}}$ signal corresponding to the currently accessed bank is manipulated by the UPM pattern when it runs. The $\overline{\text{BS}}$ signal assertion and negation timing is also specified for each cycle in the RAM word; which of the four $\overline{\text{BS}}$ signals are manipulated depends on the port size of the specified bank, the external address accessed, and the value of TSIZn. The $\overline{\text{GPL}}$ lines toggle as programmed for any access that initiates a particular pattern, but resolution of control is limited to T1 and T3.

Figure 11-58 shows how $\overline{\text{CSx}}$, $\overline{\text{GPL1}}$, and $\overline{\text{GPL2}}$ can be controlled. A word is read from the RAM that specifies on every clock cycle the logical bits CST1, CST2, CST3, CST4, G1T1, G1T3, G2T1, and G2T3. These bits indicate the electrical value for the corresponding output pins at the appropriate timing.

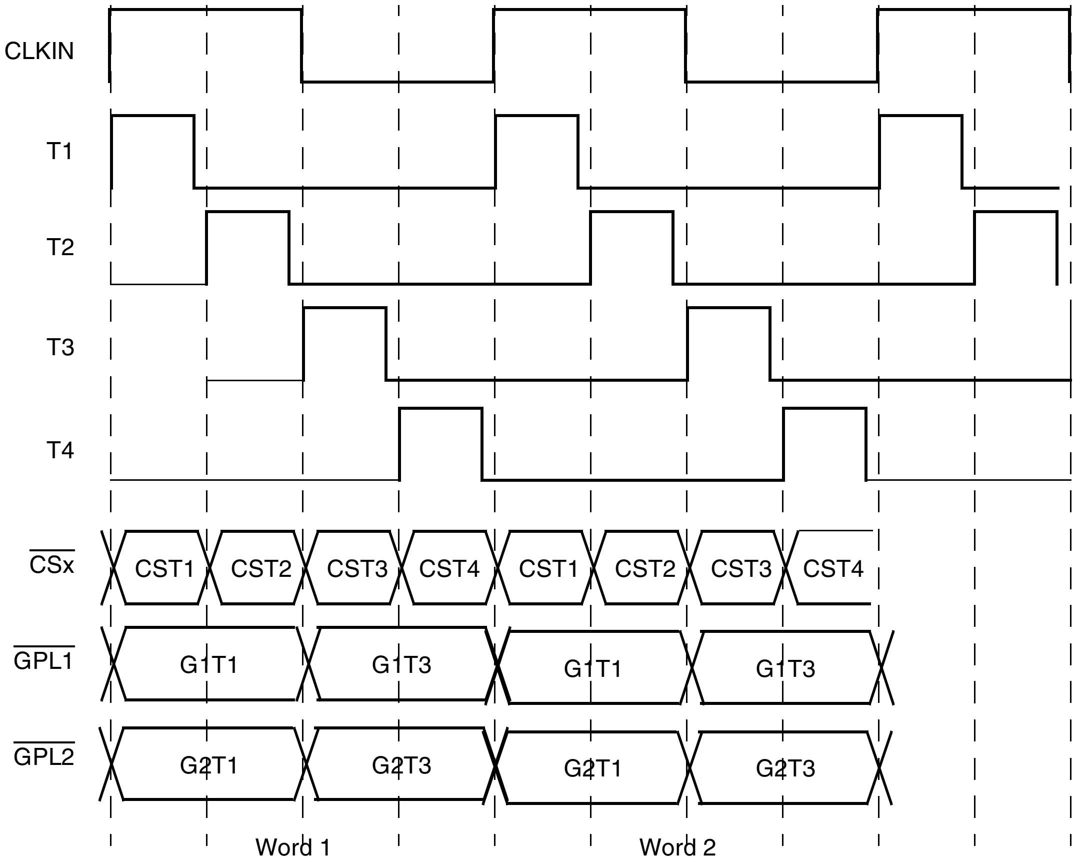


Figure 11-58. UPM Signals Timing Example

11.6.4 The RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 11-59. The signals at the bottom of Figure 11-59 are UPM outputs. The selected $\overline{\text{CS}}$ is for the bank that matches the current address. The selected $\overline{\text{BS}}$ is for the byte lanes read or written by the access.

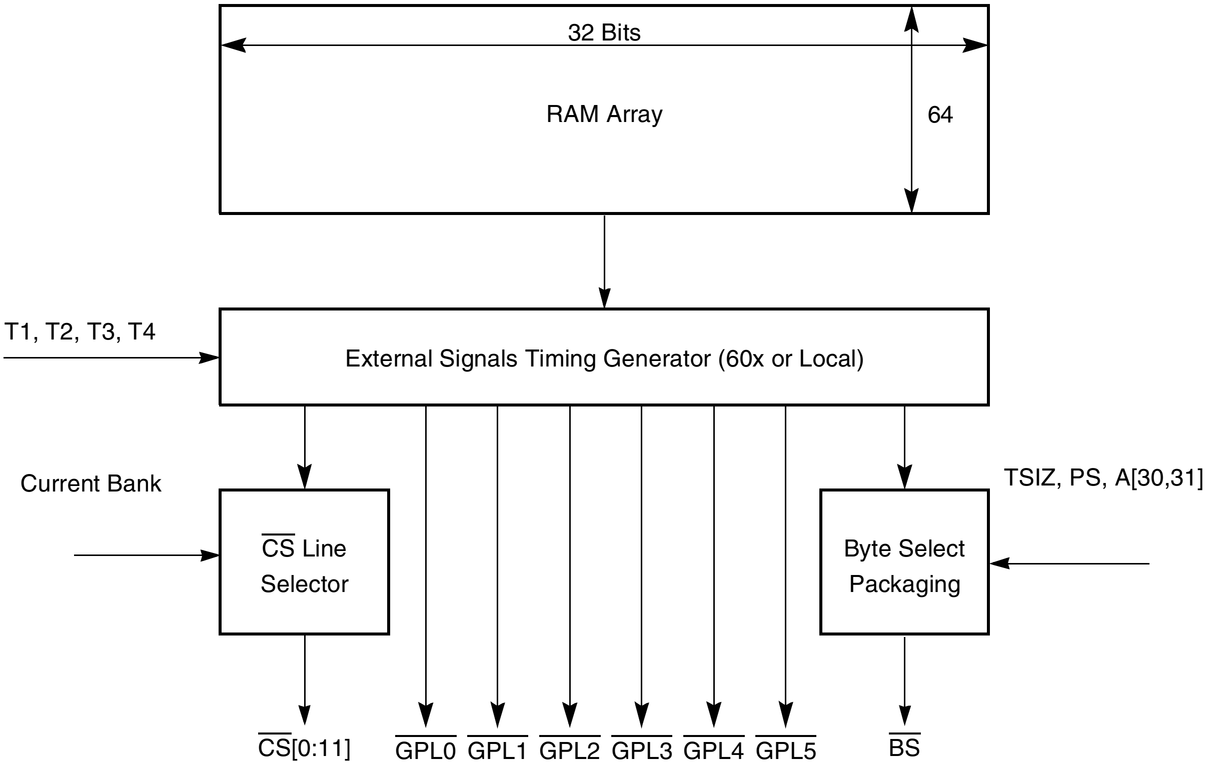


Figure 11-59. RAM Array and Signal Generation

11.6.4.1 RAM Words

The RAM word, shown in Figure 11-60, is a 32-bit micro-instruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H		G1T1	G1T3	G2T1	G2T3
Reset	—															
R/W	R/W															
Addr	(MCR[MAD] indirect addressing of 1 of 64 entries)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	G3T1	G3T3	G4T1/ DLT3	G4T3/ WAEN	G5T1	G5T3	REDO		LOOP	EXEN	AMX		NA	UTA	TODT	LAST
Reset	—															
R/W	R/W															
Addr	(All 32 bits of the RAM word are addressed as shown in the address row above.)															

Figure 11-60. The RAM Word

Table 11-33 describes RAM word fields.

Table 11-33. RAM Word Bit Settings

Bit	Name	Description
0	CST1	Chip-select timing 1. Defines the state of $\overline{\text{CS}}$ during clock phase 1. 0 The value of the $\overline{\text{CS}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.1, “Chip-Select Signals (CxTx).”
1	CST2	Chip-select timing 2. Defines the state of $\overline{\text{CS}}$ during clock phase 2. 0 The value of the $\overline{\text{CS}}$ line at the rising edge of T2 will be 0 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T2 will be 1
2	CST3	Chip-select timing 3. Defines the state of $\overline{\text{CS}}$ during clock phase 3. 0 The value of the $\overline{\text{CS}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T3 will be 1
3	CST4	Chip-select timing4. Defines the state of $\overline{\text{CS}}$ during clock phase 4. 0 The value of the $\overline{\text{CS}}$ line at the rising edge of T4 will be 0 1 The value of the $\overline{\text{CS}}$ line at the rising edge of T4 will be 1
4	BST1	Byte-select timing 1. Defines the state of $\overline{\text{BS}}$ during clock phase 1. 0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 will be 0 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 will be 1 The final value of the $\overline{\text{BS}}$ lines depends on the values of BRx[PS], the TSIZ lines, and A[30:31] for the access. See Section 11.6.4.1.2, “Byte-Select Signals (BxTx).”
5	BST2	Byte-select timing 2. Defines the state of $\overline{\text{BS}}$ during clock phase 2. 0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 will be 0 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T2 will be 1 The final value of the $\overline{\text{BS}}$ lines depends on the values of BRx[PS], TSIZ*x*, and A[30:31] for the access.
6	BST3	Byte-select timing 3. Defines the state of $\overline{\text{BS}}$ during clock phase 3. 0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T3 will be 0 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T3 will be 1 The final value of the $\overline{\text{BS}}$ lines depends on the values of BR*x*[PS], TSIZ*x*, and A[30:31] for the access.
7	BST4	Byte-select timing 4. Defines the state of $\overline{\text{BS}}$ during clock phase 4. 0 The value of the $\overline{\text{BS}}$ lines at the rising edge of T4 will be 0 1 The value of the $\overline{\text{BS}}$ lines at the rising edge of T4 will be 1 The final value of the $\overline{\text{BS}}$ lines depends on the values of BRx[PS], TSIZ*x*, and A[30:31] for the access.
8–9	G0L	General-purpose line 0 lower. Defines the state of $\overline{\text{GPL0}}$ during phases 1–2. 00 The value of $\overline{\text{GPL0}}$ at the rising edge of T1 is as defined in M*x*MR[G0CL] 10 The value of the $\overline{\text{GPL0}}$ line at the rising edge of T1 will be 0 11 The value of the $\overline{\text{GPL0}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
10–11	G0H	General-purpose line 0 higher. Defines the state of $\overline{\text{GPL0}}$ during phase 3–4. 00 The value of $\overline{\text{GPL0}}$ at the rising edge of T3 is as defined in M*x*MR[G0CL] 10 The value of the $\overline{\text{GPL0}}$ line at the rising edge of T3 will be 0 11 The value of the $\overline{\text{GPL0}}$ line at the rising edge of T3 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
12	G1T1	General-purpose line 1 timing 1. Defines the state of $\overline{\text{GPL1}}$ during phase 1–2. 0 The value of the $\overline{\text{GPL1}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{GPL1}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”

Table 11-33. RAM Word Bit Settings (continued)

Bit	Name	Description
13	G1T3	General-purpose line 1 timing 3. Defines the state of $\overline{\text{GPL1}}$ during phase 3–4. 0 The value of the $\overline{\text{GPL1}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{GPL1}}$ line at the rising edge of T3 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
14	G2T1	General-purpose line 2 timing 1. Defines the state of $\overline{\text{GPL2}}$ during phase 1–2. 0 The value of the $\overline{\text{GPL2}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{GPL2}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
15	G2T3	General-purpose line 2 timing 3. Defines the state of $\overline{\text{GPL2}}$ during phase 3–4. 0 The value of the $\overline{\text{GPL2}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{GPL2}}$ line at the rising edge of T3 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
16	G3T1	General-purpose line 3 timing 1. Defines the state of $\overline{\text{GPL3}}$ during phase 1–2. 0 The value of the $\overline{\text{GPL3}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{GPL3}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
17	G3T3	General-purpose line 3 timing 3. Defines the state of $\overline{\text{GPL3}}$ during phase 3–4. 0 The value of the $\overline{\text{GPL3}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{GPL3}}$ line at the rising edge of T3 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
18	G4T/ DLT2	General-purpose line 4 timing 1/delay time 2. The function is determined by M*x*MR[GPL*x*4DIS].
	G4T1	If M*x*MR defines UPMWAITx/$\overline{\text{GPL\_x4}}$ as an output ($\overline{\text{GPL\_x4}}$), this bit functions as G4T1: 0 The value of the $\overline{\text{GPL4}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{GPL4}}$ line at the rising edge of T1 will be 1 See Section 11.6.4.1.3, “General-Purpose Signals (GxTx, GOx).”
	DLT3	If M*x*MR[GPL*x*4DIS] = 1, UPMWAITx is chosen and this bit functions as DLT3. 0 In the current word, indicates that the data bus should be sampled at the rising edge of T1 (if a read burst or a single read service is executed). 1 In the current word, indicates that the data bus should be sampled at the rising edge of T3 (if a read burst or a single read service is executed). For an example, see Section 11.6.4.3, “Data Valid and Data Sample Control.”
19	G4T3/ WAEN	General-purpose line 4 timing 3/wait enable. Function depends on the value of M*x*MR[GPL*x*4DIS].
	G4T3	If M*x*MR[GPL*x*4DIS] = 0, G4T3 is selected. 0 The value of the $\overline{\text{GPL4}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{GPL4}}$ line at the rising edge of T3 will be 1
	WAEN	If M*x*MR[GPL*x*4DIS] = 1, WAEN is selected. See Section 11.6.4.5, “The Wait Mechanism.” 0 The UPMWAITx function is disabled. 1 A freeze in the external signal’s logical value occurs if the external wait signal is detected asserted. This condition lasts until UPMWAITx is negated.
20	G5T1	General-purpose line 5 timing 1. Defines the state of $\overline{\text{GPL5}}$ during phase 1–2. 0 The value of the $\overline{\text{GPL5}}$ line at the rising edge of T1 will be 0 1 The value of the $\overline{\text{GPL5}}$ line at the rising edge of T1 will be 1

Table 11-33. RAM Word Bit Settings (continued)

Bit	Name	Description
21	G5T3	General-purpose line 5 timing 3. Defines the state of $\overline{\text{GPL5}}$ during phase 3–4. 0 The value of the $\overline{\text{GPL5}}$ line at the rising edge of T3 will be 0 1 The value of the $\overline{\text{GPL5}}$ line at the rising edge of T3 will be 1
22–23	REDO	Redo current RAM word. See “Section 11.6.4.1.5, “Repeat Execution of Current RAM Word (REDO).” 00 Normal operation 01 The current RAM word is executed twice. 10 The current RAM word is executed tree times. 11 The current RAM word is executed four times. **Note:** For Rev A.1 and forward: for any value other than 00, do not use REDO on two consecutive CPM RAM words. The second word will not execute.
24	LOOP	Loop. 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 the start and end are defined as the loop. The number of times the UPM executes this loop is defined in the corresponding loop field of the M*x*MR. 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop. See Section 11.6.4.1.4, “Loop Control.”
25	EXEN	Exception enable. If an external device asserts $\overline{\text{TEA}}$ or $\overline{\text{RESET}}$, EXEN allows branching to an exception pattern at the exception start address (EXS) at a fixed address in the RAM array. When the MPC8272 under UPM control begins accessing a memory device, the external device may assert $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$. An exception occurs when one of these signals is asserted by an external device and the MPC8272 begins closing the memory cycle transfer. When one of these exceptions 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. See Table 11-32.. The user should provide an exception pattern to deassert signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ to prevent data corruption. If EXEN = 0, exceptions are deferred 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. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected. The exception condition can be an external device asserting $\overline{\text{TEA}}$ or $\overline{\text{SRESET}}$.
26–27	AMX	Address multiplexing. Determines the source of A[0–31] at the rising edge of t1 (single-MPC8272 mode only). See Section 11.6.4.2, “Address Multiplexing.” 00 A[0–31] is the non-multiplexed address. For example, column address. 01 Reserved. 10 A[0–31] is the address requested by the internal master multiplexed according to M*x*MR[AM*x*]. For example, row address. 11 A[0–31] is the contents of MAR. Used for example, during SDRAM mode initialization.
28	NA	Next 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*x*[PS], the increment value of A[27–31] and/or BADDR[27–31] at the rising edge of T1 is as follows If the accessed bank has a 64-bit port size, the value is incremented by 8. If the accessed bank has a 32-bit port size, the value is incremented by 4. If the accessed bank has a 16-bit port size, the value is incremented by 2. If the accessed bank has an 8-bit port size, the value is incremented by 1. **Note:** The value of NA is relevant only when the UPM serves a burst-read or burst-write request. NA is reserved under other patterns.

Table 11-33. RAM Word Bit Settings (continued)

Bit	Name	Description
29	UTA	UPM transfer acknowledge. Indicates assertion of $\overline{\mathrm{PSDVAL}}$, sampled by the bus interface in the current cycle. 0 $\overline{\mathrm{PSDVAL}}$ is not asserted in the current cycle. 1 $\overline{\mathrm{PSDVAL}}$ is asserted in the current cycle.
30	TODT	Turn-on disable timer. The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a $\overline{\mathrm{RAS}}$ precharge time. TODT, turns the timer on to prevent another UPM access to the same bank until the timer expires.The disable timer period is determined in M*x*MR[DS*x*]. The disable timer does not affect memory accesses to different banks. 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. **Note:** TODT must be set together with LAST. Otherwise it is ignored.
31	LAST	Last. If this bit is set, it is the last RAM word in the program. When the LAST bit is read in a RAM word, the current UPM pattern terminates and the highest priority pending UPM request (if any) is serviced immediately in the external memory transactions. If the disable timer is activated and the next access is top the same bank, the execution of the next UPM pattern is held off for the number of clock cycles specified in MxMR[DSx]. 0 The UPM continues executing RAM words. 1 The service to the UPM request is done.

Additional information about some of the RAM word fields is provided in the following sections.

11.6.4.1.1 Chip-Select Signals (C*x*T*x*)

If BRx[MS] of the accessed bank selects a UPM on the currently requested cycle the UPM manipulates the $\overline{\mathrm{CS}}$ signal for that bank with timing as specified in the UPM RAM word. The selected UPM affects only assertion and negation of the appropriate $\overline{\mathrm{CS}}$ signal. The state of the selected $\overline{\mathrm{CSx}}$ signal of the corresponding bank depends on the value of each CSTn bit.

Figure 11-61 and the timing diagrams in Figure 11-58 show how UPMs control $\overline{\mathrm{CS}}$ signals.

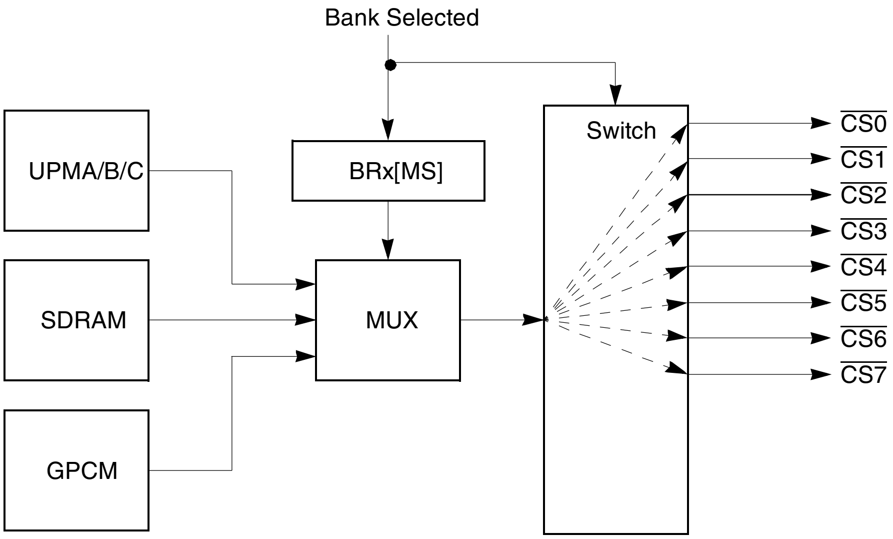


Figure 11-61. CS Signal Selection

11.6.4.1.2 Byte-Select Signals (B*x*T*x*)

BRx[MS] of the accessed memory bank selects a UPM on the currently requested cycle. The selected UPM affects only the assertion and negation of the appropriate BS signal; its timing as specified in the RAM word. The BS signals are controlled by the port size of the accessed bank, the transfer size of the transaction, and the address accessed. Figure 11-62 shows how UPMs control BS signals.

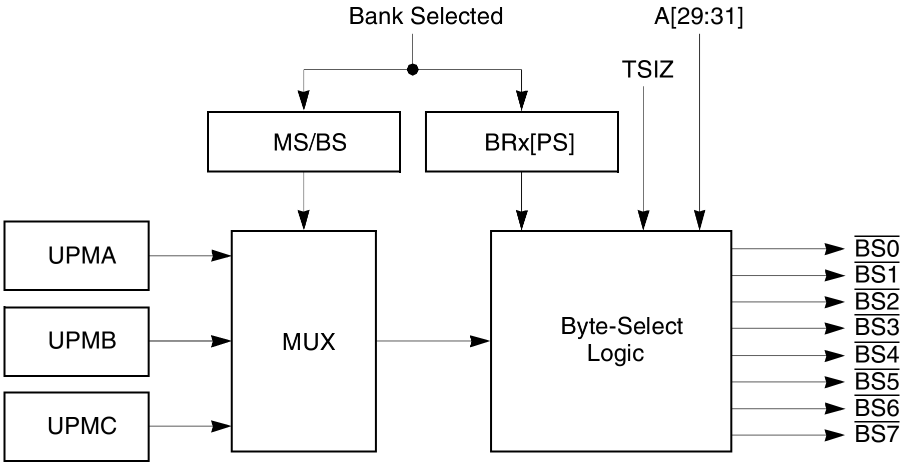


Figure 11-62. BS Signal Selection

The uppermost byte select (BS0) indicates that D[0–7] contains valid data during a cycle. Likewise, BS1 indicates that D[8–15] contains valid data, BS2 indicates that D[16–23] contains valid data, and BS3 indicates that D[24–31] contains valid data during a cycle, and so forth. Note that for a refresh timer request, all the BS signals are asserted/negated by the UPM.

11.6.4.1.3 General-Purpose Signals (G*x*T*x*, GO*x*)

The general-purpose signals ($\overline{\text{GPL}}$[1:5]) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of T1 and/or at the rising edge of T3. $\overline{\text{GPL0}}$ offer enhancements beyond the other $\overline{\text{GPLx}}$ lines.

$\overline{\text{GPL0}}$ can be controlled by an address line specified in MxMR[G0CLx]. To use this feature, set G0H and G0L in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between banks.

11.6.4.1.4 Loop Control

The LOOP bit in the RAM word (bit 24) 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 11-34.. 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.

Table 11-34. M*x*MR Loop Field Usage

Request Serviced	Loop Field
Read single-beat cycle	RLFx
Read burst cycle	RLFx
Write single-beat cycle	WLFx
Write burst cycle	WLFx
Refresh timer expired	TLFx
RUN command	RLFx

11.6.4.1.5 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value to cause the UPM to re-execute the current RAM word up to three times, according to Table 11-33.

Special care must be taken in the following cases:

- When UTA and REDO are set together, $\overline{\text{PSDVAL}}$ is asserted 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 should not be set together.
- REDO should not be used within the exception routine.

Figure 11-77 on page 11-88 shows an example of REDO use.

11.6.4.2 Address Multiplexing

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between outputting an address requested by the internal master as is and outputting it according to the multiplexing specified by the MxMR[AMx]. The last option is to output the contents of the MAR on the address pins.

Note that in 60x-compatible mode, MAR cannot be output on the 60x bus external address line. Also note that for the UPM address multiplexing to work, PSDMR[PBI] must be cleared.

Table 11-35 shows how MxMR[AMx] settings affect address multiplexing.

Table 11-35. UPM Address Multiplexing

AM*x*	External Bus Address Pin	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	**Signal Driven on External Pin when Address Multiplexing is Enabled**	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		—	—	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

See Section 11.6.5, “UPM DRAM Configuration Example,” for more details.

11.6.4.3 Data Valid and Data Sample Control

When a read access is handled by the UPM and the UTA bit is 1, the value of the DLT3 bit in the same RAM word indicates when the data input is sampled by the internal bus master, assuming that MxMR[GPLx4DIS] = 1.

- If G4T4/DLT3 functions as DLT3 and DLT3 = 1 in the RAM word, data is latched on the falling edge of CLKIN instead of the rising edge. The data is sampled by the internal master on the next rising edge as is required by the MPC8272 bus operation spec. This feature lets the user speed up the memory interface by latching data 1/2 clock early, which can be useful during burst reads. This feature should be used only in systems without external synchronous bus devices.
- If G4T4/DLT3 functions as G4T4, data is latched on the rising edge of CLKIN, as is normal in MPC8272 bus operation.

Figure 11-63 shows data sampling that is controlled by the UPM.

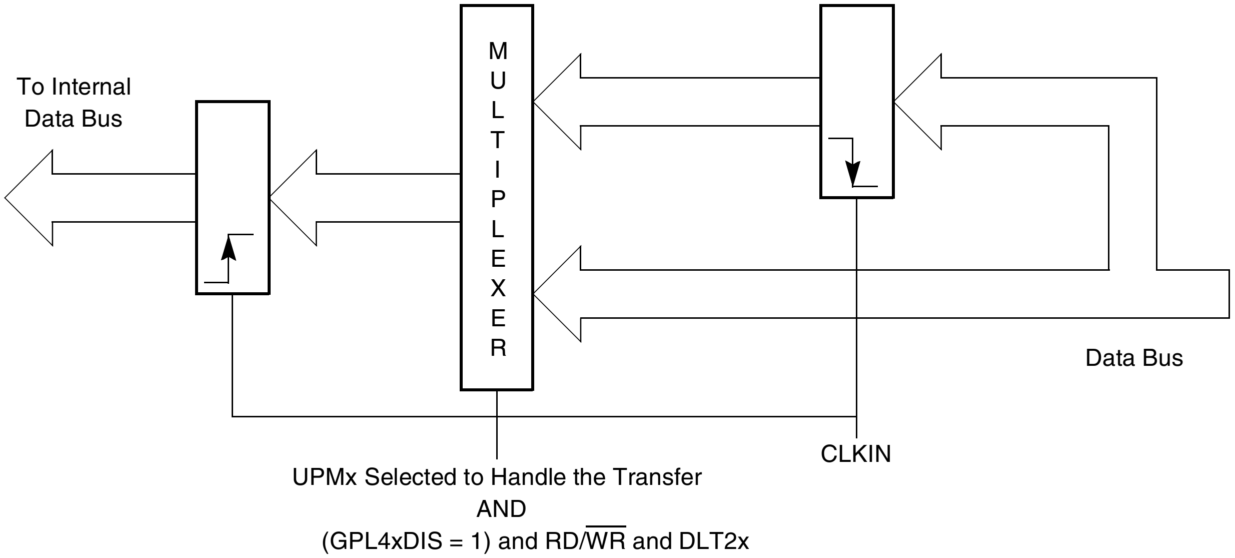

To Internal
Data Bus
M
U
L
T
I
P
L
E
X
E
R
Data Bus
CLKIN
UPMx Selected to Handle the Transfer
AND
(GPL4xDIS = 1) and RD/$\overline{WR}$ and DLT2x

Figure 11-63. UPM Read Access Data Sampling

11.6.4.4 Signals Negation

When the LAST bit is read in a RAM word, the current UPM pattern terminates. On the next cycle all the UPM signals are negated unconditionally (driven to logic '1').

This negation will not occur only if there is a back-to-back UPM request pending. In this case the signals value on the cycle following the LAST bit, will be taken from the first line of the pending UPM routine.

11.6.4.5 The Wait Mechanism

The WAEN bit in the RAM array word, shown in Table 11-33, can be used to enable the UPM wait mechanism in selected UPM RAM words. Note that the WAEN bit needs to be set in two consecutive UPM words to get the desired operation.

If the UPM reads a RAM word with the WAEN bit set, the external UPMWAIT signal is sampled by the memory controller in the following cycle and the request is frozen. The UPMWAIT signal is sampled at the rising edge of CLKIN. If UPMWAIT is asserted and WAEN = 1 in the previous UPM word, the UPM is frozen until UPMWAIT is negated. The value of the external pins driven by the UPM remains as indicated in the previous word read by the UPM. When UPMWAIT is negated, the UPM continues its normal functions. Note that during the wait cycles, the UPM negates $\overline{\text{PSDVAL}}$.

Figure 11-64 shows how the WAEN bit in the word read by the UPM and the UPMWAIT signal are used to hold the UPM in a particular state until UPMWAIT is negated. As the example in Figure 11-64 shows, the $\overline{\text{CSx}}$ and $\overline{\text{GPL1}}$ states (C12 and F) and the WAEN value (C) are frozen until UPMWAIT is recognized as deasserted. WAEN is typically set before the line that contain UTA = 1.

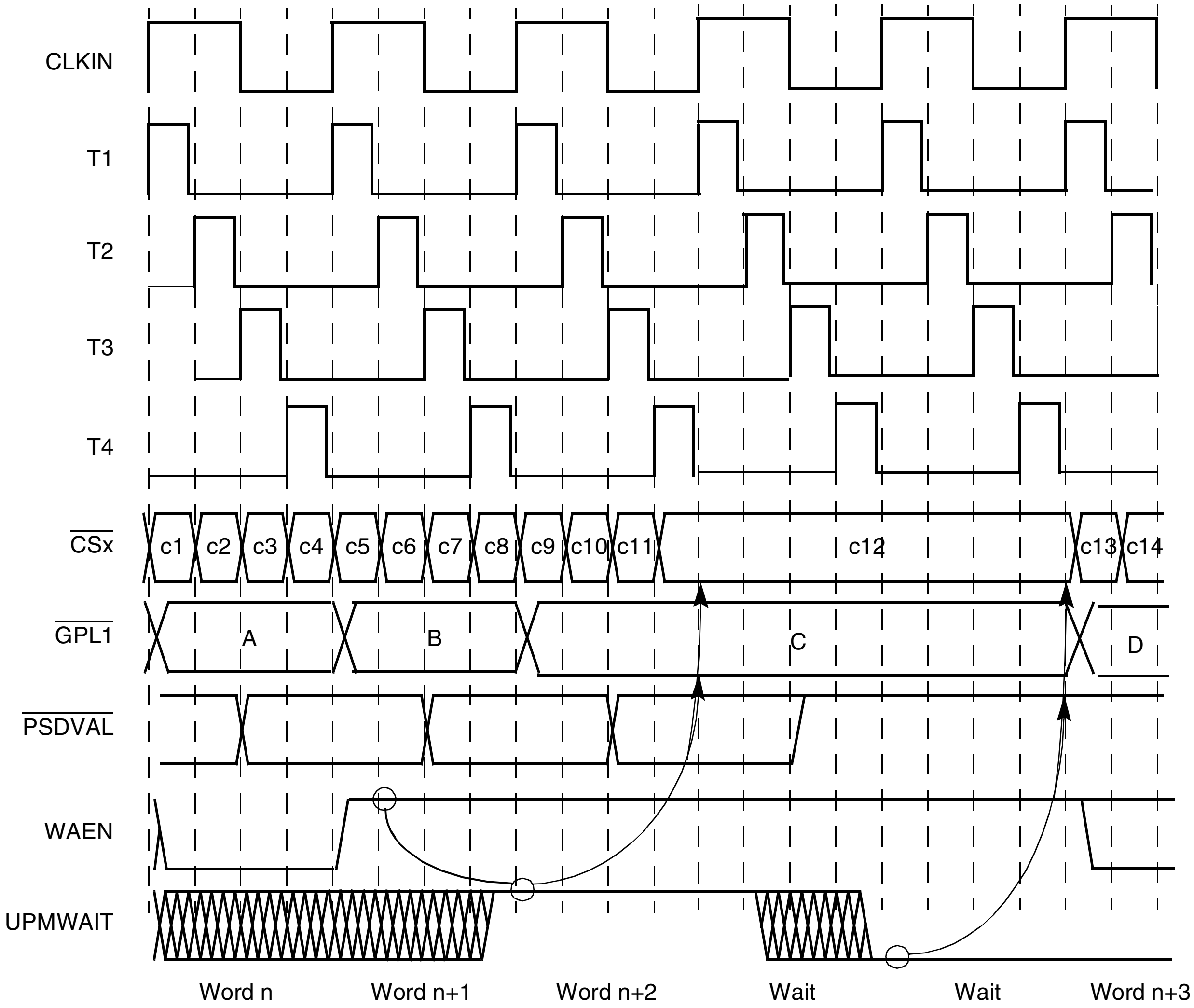


Figure 11-64. Wait Mechanism Timing for Internal and External Synchronous Masters

11.6.4.6 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 chose some combination of ORx[EHTR]. Accesses after a read access to the slower memory bank is delayed by the number of clock cycles specified by Table 11-29. The information in Section 11.5.1.6, “Extended Hold Time on Read Accesses,” provides additional information.

11.6.5 UPM DRAM Configuration Example

\Consider the following DRAM organization:

- Eight 64Mbit devices, each organized as 8M x 8bits
- Each device has 12 row lines and 9 column lines.

This means that the address bus should be partitioned as shown in Table 11-36.

Table 11-36. 60x Address Bus Partitions

A[0:7]	A[8:19]	A[20:28]	A[29:31]
msb of start address	Row	Column	lsb

From the device perspective, during $\overline{\text{RAS}}$ assertion, its address port should look like Table 11-37:

Table 11-37. DRAM Device Address Port during an ACTIVATE Command

A[0:16]	A[17:28]	A[29:31]
—	Row (A[8:19])	n.c.

Table 11-35 indicates that to multiplex A[8:19] over A[17:28], choose AMx = 001.

Table 11-38 shows the register configuration. Not shown are PURT and MPTPR, which should be programmed according to the device refresh requirements.

Table 11-38. Register Settings

Register	Settings	
BRx	BA msb of base address PS 00 = 64-bit port size DECC00 WP0 MS 100 = UPMA	EMEMC0 ATOM00 DR 0 V 1
OR*x*	AM 1111_1111_0000_0000_0 = 16 Mbyte BI 0	EHTR0
M*x*MR	BSEL0 = 60x bus RFEN1 OP 00 AM 001 DSAAs needed G0CLAN/A	GPL_A4DIS0 RLFAAs needed WLFAAs needed TLFAAs needed MADN/A

11.6.6 Differences between MPC8xx UPM and MPC82xx UPM

Users familiar with the MPC8xx UPM should read this section first.

Below is a list of the major differences between the MPC8xx devices and the MPC82xx:

- First cycle timing transferred to the UPM array—In the MPC8xx's UPM, the first cycle value of some of the signals is determined from ORx[SAM,G5LA,G5LS]. This is eliminated in the MPC8272. All signals are controlled only by the pattern written to the array.
- Timing of GPL[0:5]—In the MPC8xx's UPM, the GPL lines could change on the positive edge of T2 or T3. In the MPC8272 these signals can change in the positive edge of T1 or T3 to allow connection to high-speed synchronous devices such as burst SRAM.
- UPM controlled signals negated at end of an access—In the MPC8xx's UPM, if the user did not negate the UPM signals at end of an access, those signals kept their previous value. In the

PowerQUICC II, all UPM signals are negated ($\overline{CS}$,$\overline{BS}$,GPL[0:4] driven to logic 1 and GPL5 driven to logic 0) at the end of that cycle, unless there is a back-to-back UPM cycle pending. In many cases this allows the UPM routine to finish one cycle earlier because it is now possible and desired to assert both UTA and LAST.

- MCR is eliminated—In the MPC8272, MCR is eliminated. The function of RAM read/write and RUN is done through the MxMR.
- UTA polarity is reversed—In the MPC8272, UTA is active high.
- The disable timer control (TODT) and LAST bit in the RAM array word must be set together, otherwise TODT is ignored.
- Refresh timer value is in a separate register—In the MPC8272, the refresh timer value has moved to two registers, PURT and LURT, which can serve multiple UPMs.
- Refresh on the 60x bus must be done in UPMA.
- New feature: Repeated execution of the current RAM word (REDO).
- Extended hold time on reads can be up to 8 clock cycles instead of 1 in the MPC8xx.
- Each UPM on the MPC8xx has a wait signal. On the MPC8272, the three UPMs share two wait signals (PUPMWAIT and LUPMWAIT).

11.7 Memory System Interface Example Using UPM

Connecting the MPC8272 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 provides timing diagrams for various UPM configurations.

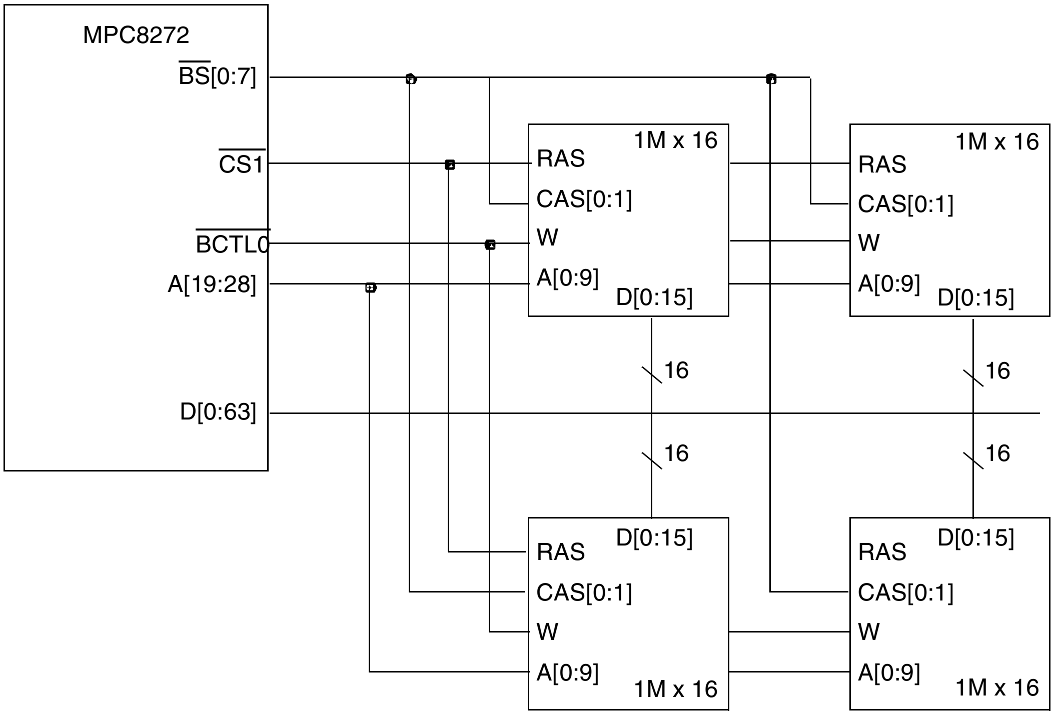


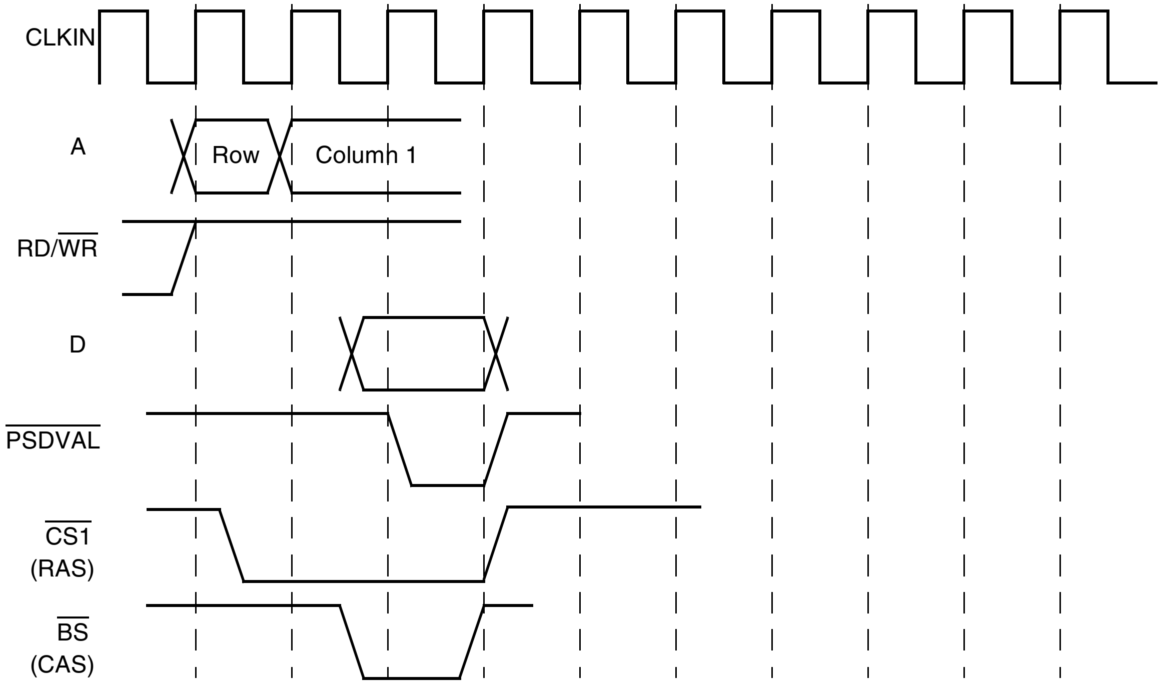
Figure 11-65. DRAM Interface Connection to the 60x Bus (64-Bit Port Size)

After timings are created, programming the UPM continues with translating these timings into tables representing the RAM array contents for each possible cycle. When a table is completed, the global parameters of the UPM must be defined for handling the disable timer (precharge) and the refresh timer relative to Figure 11-65. Table 11-39 shows settings of different fields.

Table 11-39. UPMs Attributes Example

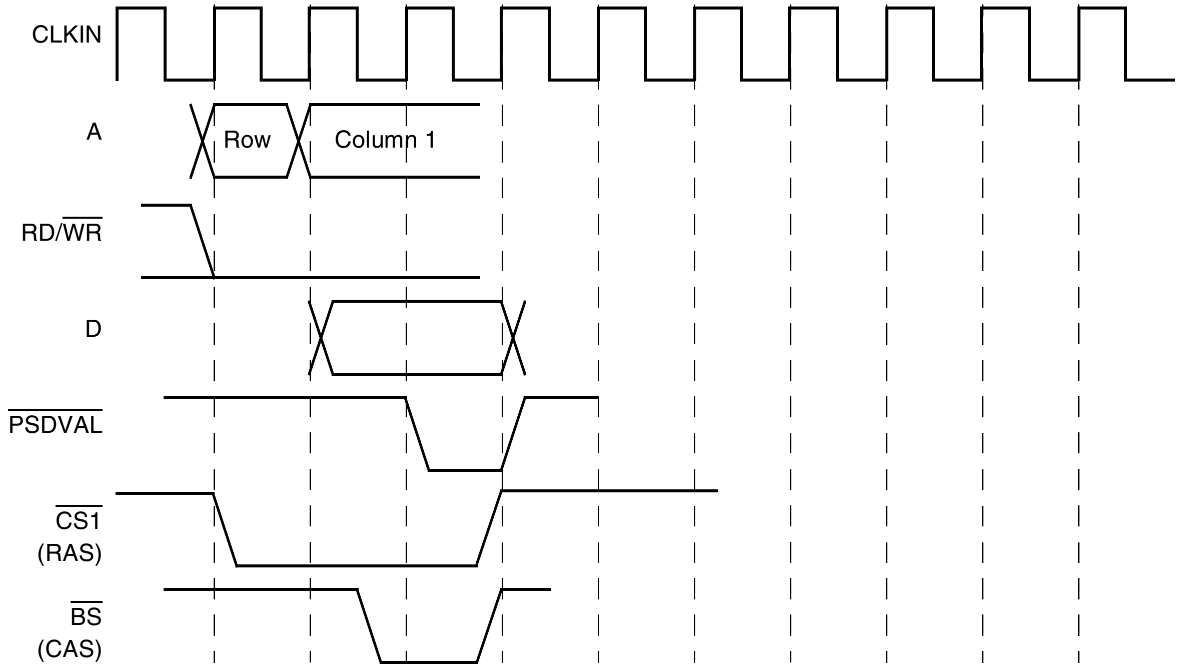
Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect (R/W)	BR*x*[WP]	0b0
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x0C
Refresh timer enable	M*x*MR[RFEN]	0b1
Address multiplex size	M*x*MR[AM*x*]	0b010
Disable timer period	M*x*MR[DS*x*]	0b01
Select between GPL4 and UPMWAIT = GPL4 data sample at clock rising edge	M*x*MR[GPL_x4DIS]	0b0
Burst inhibit device	OR*x*[BI]	0b0

The OR and BR of the specific bank must be initialized according to the address mapping of the DRAM device used. The MS field should indicate the specific UPM selected to handle the cycle. The RAM array of the UPM can than be written through use of the MxMR[OP] = 11. Figure 11-54 shows the first locations addressed by the UPM, according to the different services required by the DRAM.



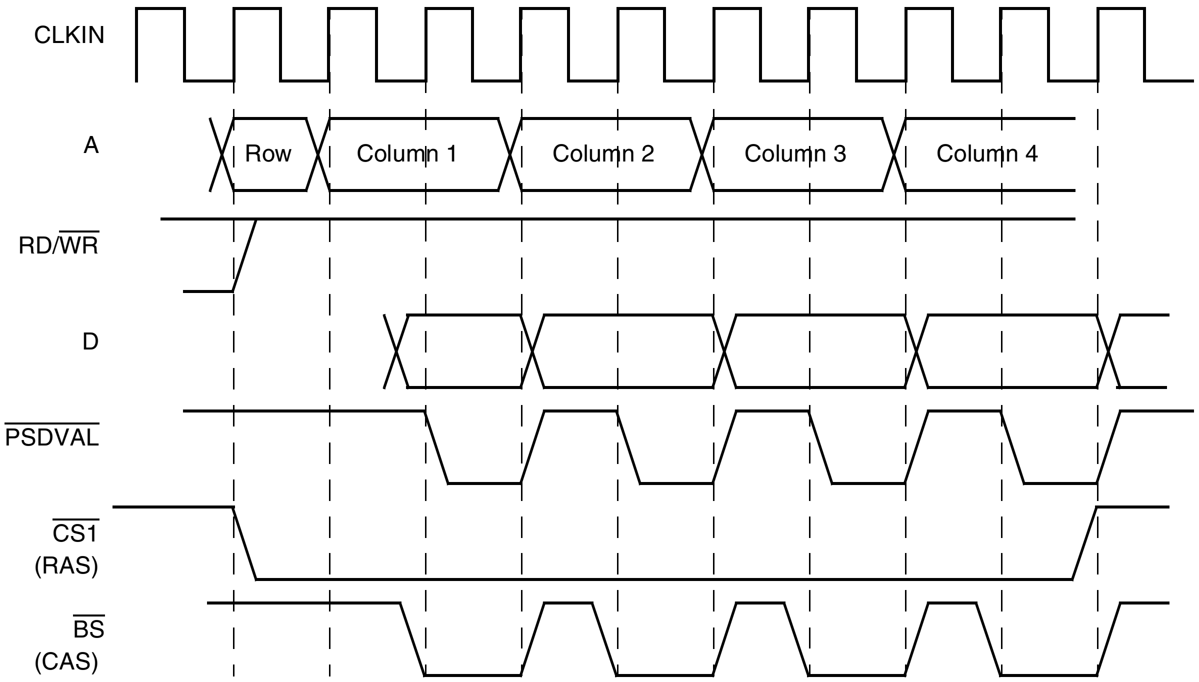
cst1	0	0	0		Bit 0
cst2	0	0	0		Bit 1
cst3	0	0	0		Bit 2
cst4	0	0	0		Bit 3
bst1	1	1	0		Bit 4
bst2	1	0	0		Bit 5
bst3	1	0	0		Bit 6
bst4	1	0	0		Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1					Bit 12
g1t3					Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
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	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+ 2**		

Figure 11-66. Single-Beat Read Access to FPM DRAM



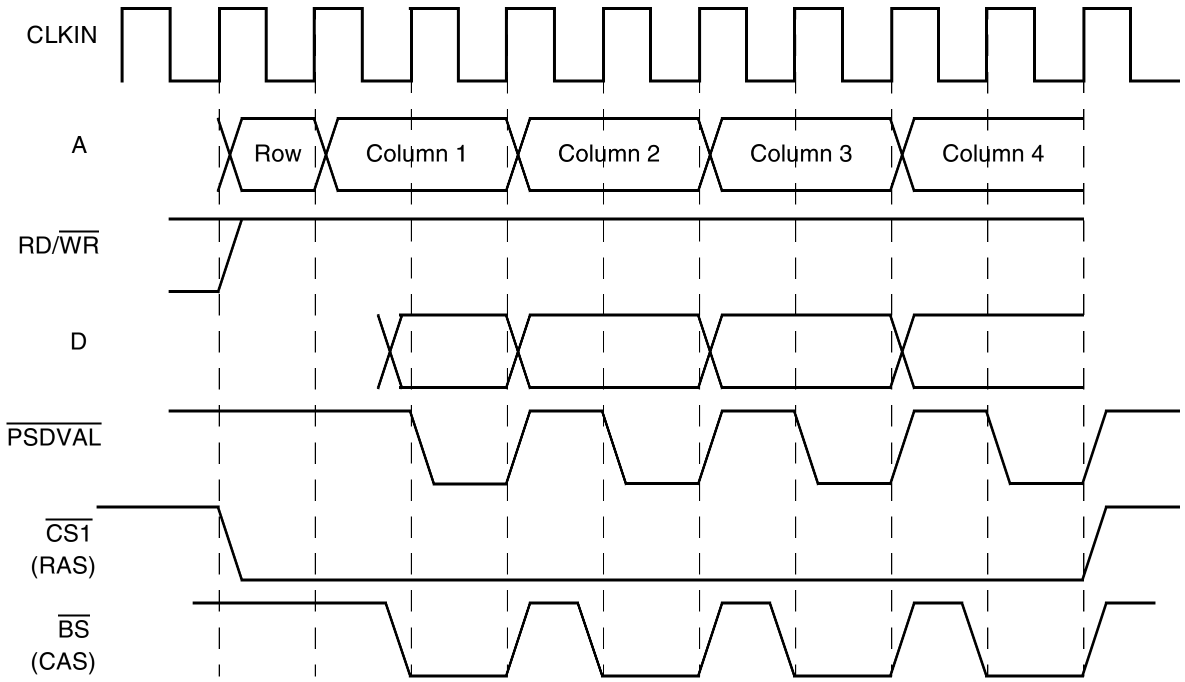
cst1	0	0	0		Bit 0
cst2	0	0	0		Bit 1
cst3	0	0	0		Bit 2
cst4	0	0	1		Bit 3
bst1	1	1	0		Bit 4
bst2	1	1	0		Bit 5
bst3	1	0	0		Bit 6
bst4	1	0	1		Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1					Bit 12
g1t3					Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
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	1	0	0		Bit 26
amx1	0	0	0		Bit 27
na	0	0	0		Bit 28
uta	0	0	1		Bit 29
todt	0	0	1		Bit 30
last	0	0	1		Bit 31
	WSS	**WSS+ 1**	**WSS+ 2**		

Figure 11-67. Single-Beat Write Access to FPM DRAM



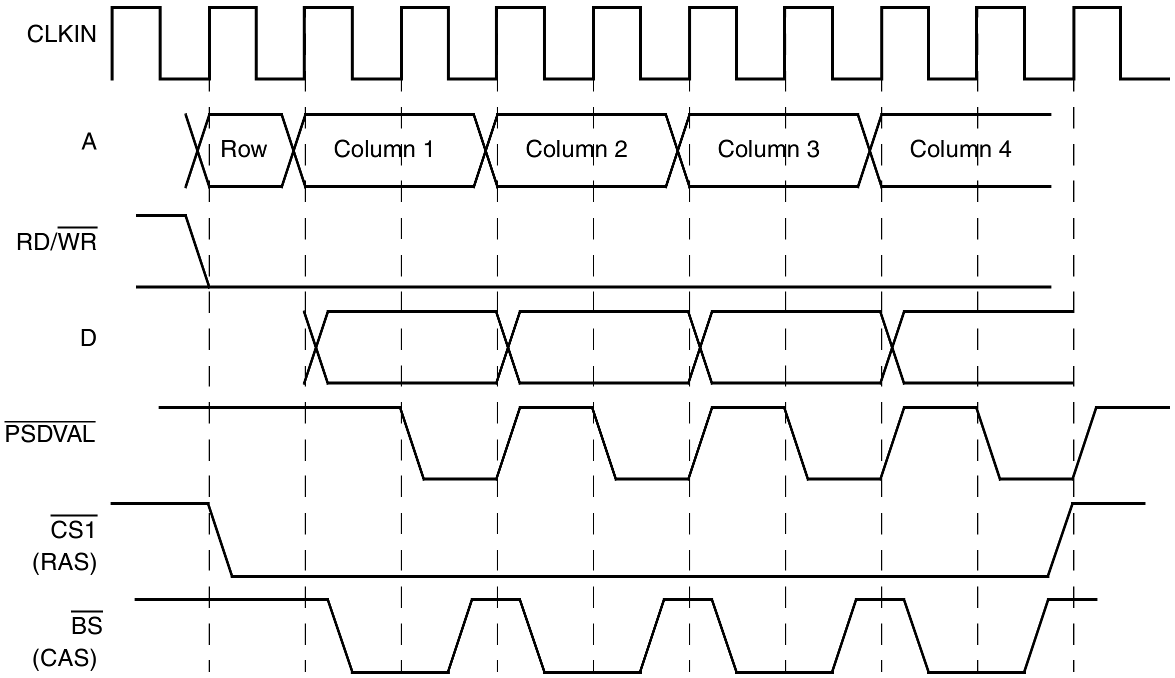
cst1 cst2 cst3 cst4	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1		Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	1 1 1 0	0 0 0 0	1 1 1 0	0 0 0 0	1 1 1 0	0 0 0 0	1 1 1 0	0 0 0 0		Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1											Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3											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 1	0 0	0 1	0 0	0 1	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 1 1 0 0	0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 0 0	0 0 1 1 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
	RBS	**RBS+ 1**	**RBS+ 2**	**RBS+ 3**	**RBS+ 4**	**RBS+ 5**	**RBS+ 6**	**RBS+ 7**	**RBS+ 8**		

Figure 11-68. Burst Read Access to FPM DRAM (No LOOP)



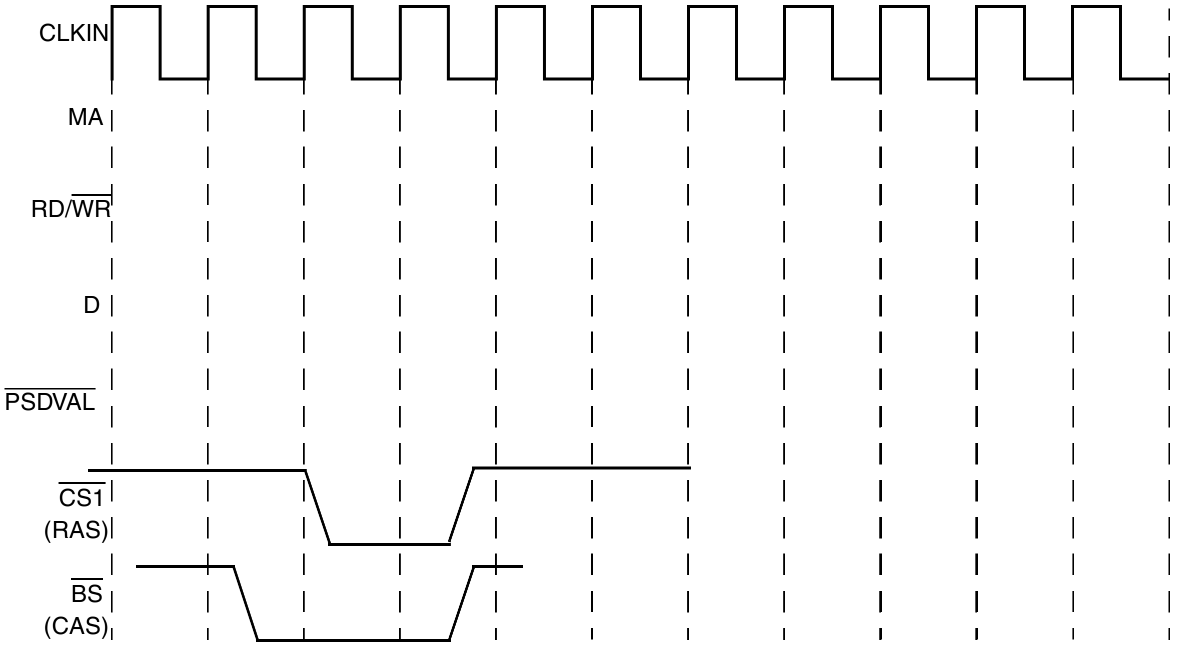
cst1	0	0	0				Bit 0
cst2	0	0	0				Bit 1
cst3	0	0	0				Bit 2
cst4	0	0	0				Bit 3
bst1	1	1	0				Bit 4
bst2	1	1	0				Bit 5
bst3	1	1	0				Bit 6
bst4	1	0	0				Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1							Bit 12
g1t3							Bit 13
g2t1							Bit 14
g2t3							Bit 15
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	1	1				Bit 24
exen	0	0	1				Bit 25
amx0	1	0	0				Bit 26
amx1	0	0	0				Bit 27
na	0	0	1				Bit 28
uta	0	0	1				Bit 29
todt	0	0	1				Bit 30
last	0	0	1				Bit 31
	RBS	**RBS+ 1**	**RBS+ 2**	**RBS+ 3**	**RBS+ 4**		

Figure 11-69. Burst Read Access to FPM DRAM (LOOP)



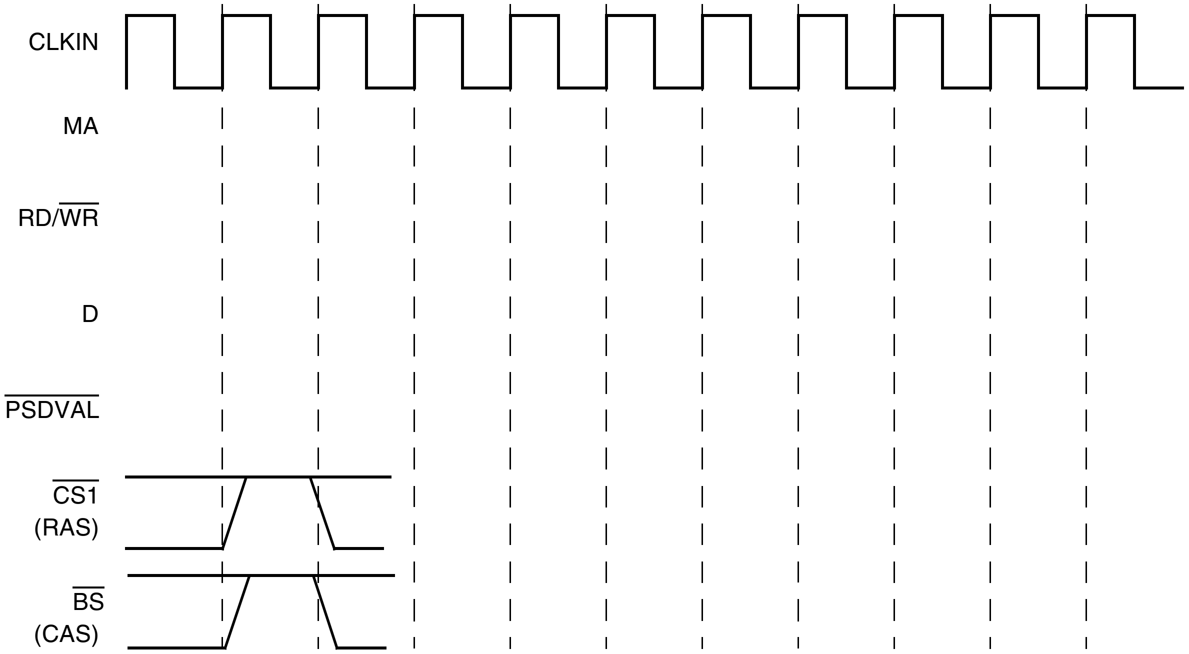
cst1 cst2 cst3 cst4	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1		Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	1 0 0 0	0 0 1 1	1 0 0 0	0 0 1 1	1 0 0 0	0 0 1 1	1 0 0 0	0 0 1 1		Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1											Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3											Bit 12 Bit 13
g2t1 g2t3											Bit 14 Bit 15
g3t1 g3t3											Bit 16 Bit 17
g4t1 g4t3											Bit 18 Bit 19
g5t1 g5t3											Bit 20 Bit 21
redo[0] redo[1]											Bit 22 Bit 23
loop exen	0 0	0 0	0 1	0 0	0 1	0 0	0 1	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 1 1 0 0	0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 0 0	0 0 1 1 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
	WBS	**WBS+ 1**	**WBS+ 2**	**WBS+ 3**	**WBS+ 4**	**WBS+ 5**	**WBS+ 6**	**WBS+ 7**	**WBS+ 8**		

Figure 11-70. Burst Write Access to FPM DRAM (No LOOP)



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 11-71. 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 11-72. Exception Cycle

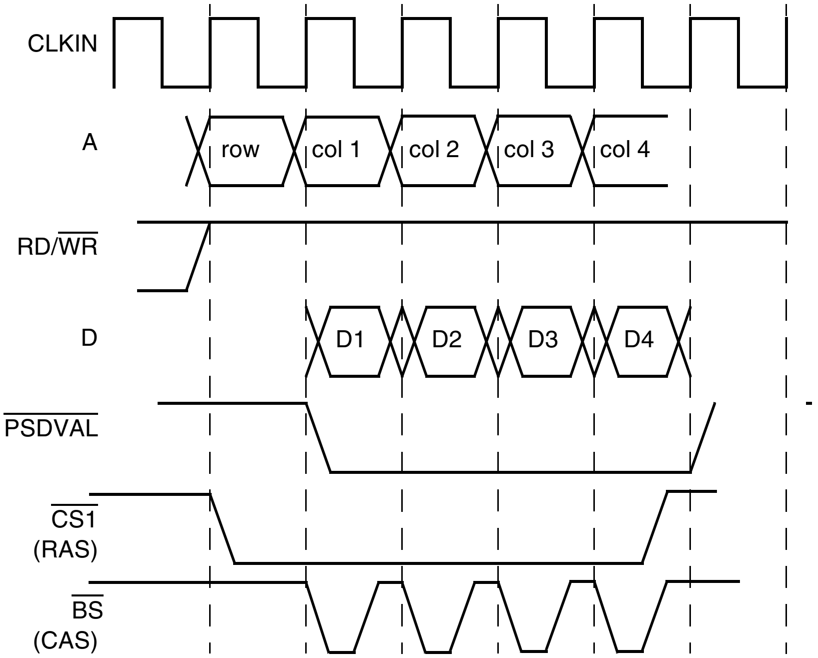
- If $\overline{\text{GPL\_4}}$ is not used as an output, the performance for a page read access can be improved by setting MxMR[GPL_x4DIS]. The following example shows how the burst read access to FPM

DRAM (no LOOP) can be modified using this feature. In this case the configuration registers are defined in the following way.

Table 11-40. UPMs Attributes Example

Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect (R/W)	BR*x*[WP]	0b0
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x0C
Refresh timer enable	M*x*MR[RFEN]	0b1
Address multiplex size	M*x*MR[AM*x*]	0b010
Disable timer period	M*x*MR[DS*x*]	0b01
Select between GPL4 and UPMWAIT = UPMWAIT, data sampled at clock negative edge	M*x*MR[GPL_x4DIS]	0b1
Burst inhibit device	OR*x*[BI]	0b0

The timing diagram in Figure 11-73 shows how the burst-read access shown in Figure 11-68 can be reduced.



cst1	0	0	0	0	0			Bit 0
cst2	0	0	0	0	0			Bit 1
cst3	0	0	0	0	1			Bit 2
cst4	0	0	0	0	1			Bit 3
bst1	1	0	0	0	0			Bit 4
bst2	1	0	0	0	0			Bit 5
bst3	1	1	1	1	1			Bit 6
bst4	1	1	1	1	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 -> DLT3	1	1	1	1	1			Bit 18
g4t3	0	0	0	0	0			Bit 19
g5t1								Bit 20
g5t3								Bit 21
redo[0]								Bit 22
redo[1]								Bit 23
loop	0	0	0	0	0			Bit 24
exen	0	0	0	0	0			Bit 25
amx0	1	0	0	0	0			Bit 26
amx1	0	0	0	0	0			Bit 27
na	0	1	1	1	0			Bit 28
uta	0	1	1	1	1			Bit 29
todt	0	0	0	0	1			Bit 30
last	0	0	0	0	1			Bit 31
	RBS	**RBS+ 1**	**RBS+ 2**	**RBS+ 3**	**RBS+ 4**	**RBS+ 5**		

Figure 11-73. FPM DRAM Burst Read Access (Data Sampling on Falling Edge of CLKIN)

11.7.0.1 EDO Interface Example

Figure 11-74 shows a memory connection to extended data-out type devices. For this connection, $\overline{\text{GPL1}}$ is connected to the memory device’s $\overline{\text{OE}}$ pins.

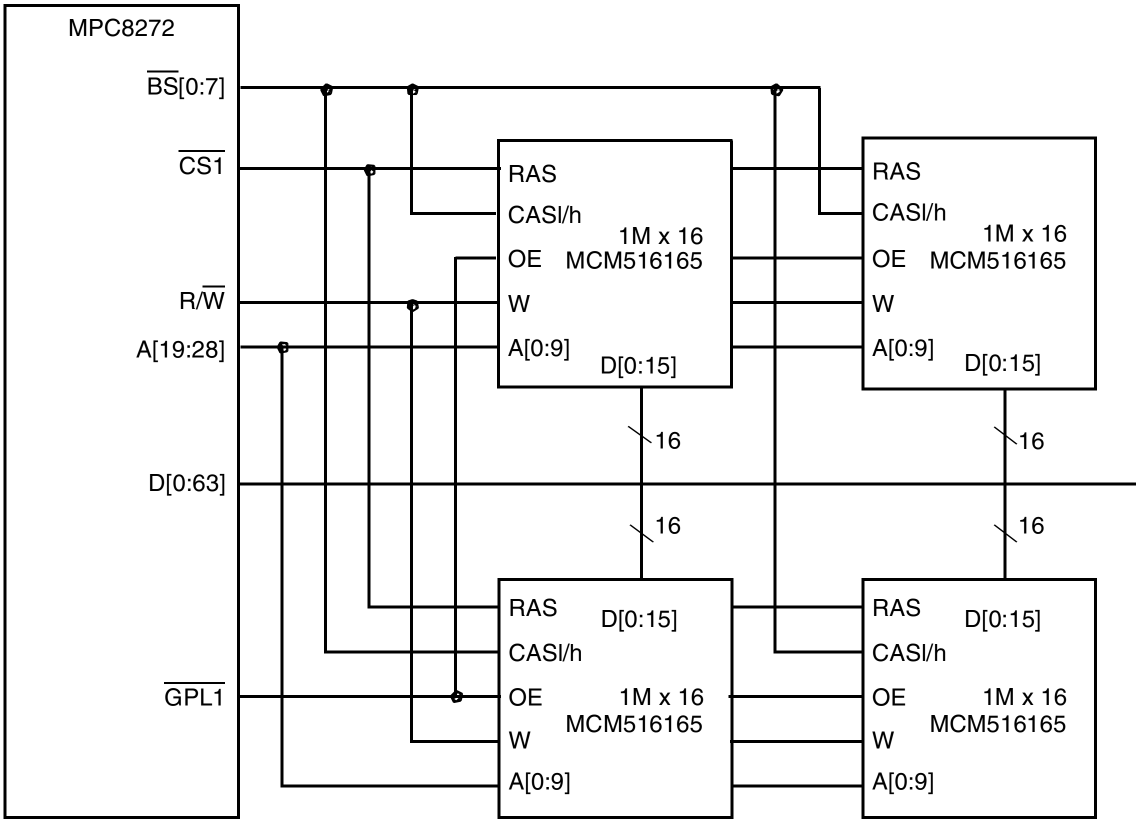
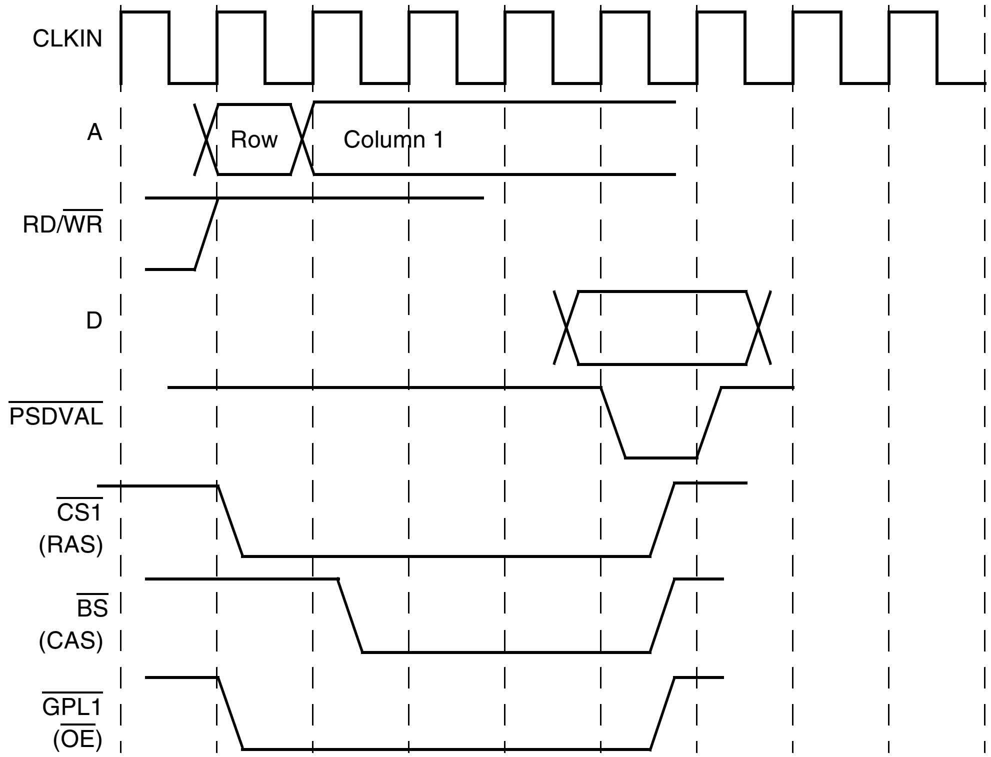


Figure 11-74. MPC8272/EDO Interface Connection to the 60x Bus

Table 11-41 shows the programming of the register field for supporting the configuration shown in Figure 11-74. The example assumes a CLKIN frequency of 66 MHz and that the device needs a 1,024-cycle refresh every 10 µs.

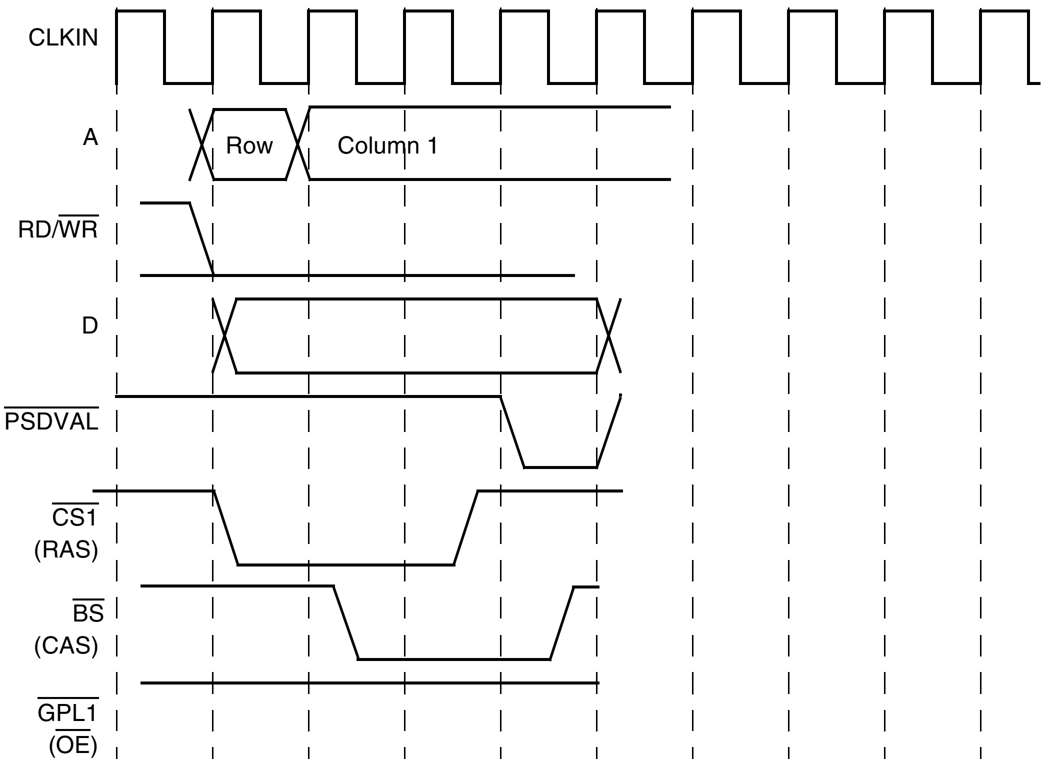
Table 11-41. EDO Connection Field Value Example

Explanation	Field	Value
Machine select UPMA	BR*x*[MS]	0b100
Port size 64-bit	BR*x*[PS]	0b00
No write protect (R/W)	BR*x*[WP]	0b0
Refresh timer prescaler	MPTPR	0x04
Refresh timer value (1024 refresh cycles)	PURT[PURT]	0x07
Refresh timer enable	M*x*MR[RFEN]	0b1
Address multiplex size	M*x*MR[AM*x*]	0b001
Disable timer period	M*x*MR[DS*x*]	0b10
Burst inhibit device	OR*x*[BI]	0b0



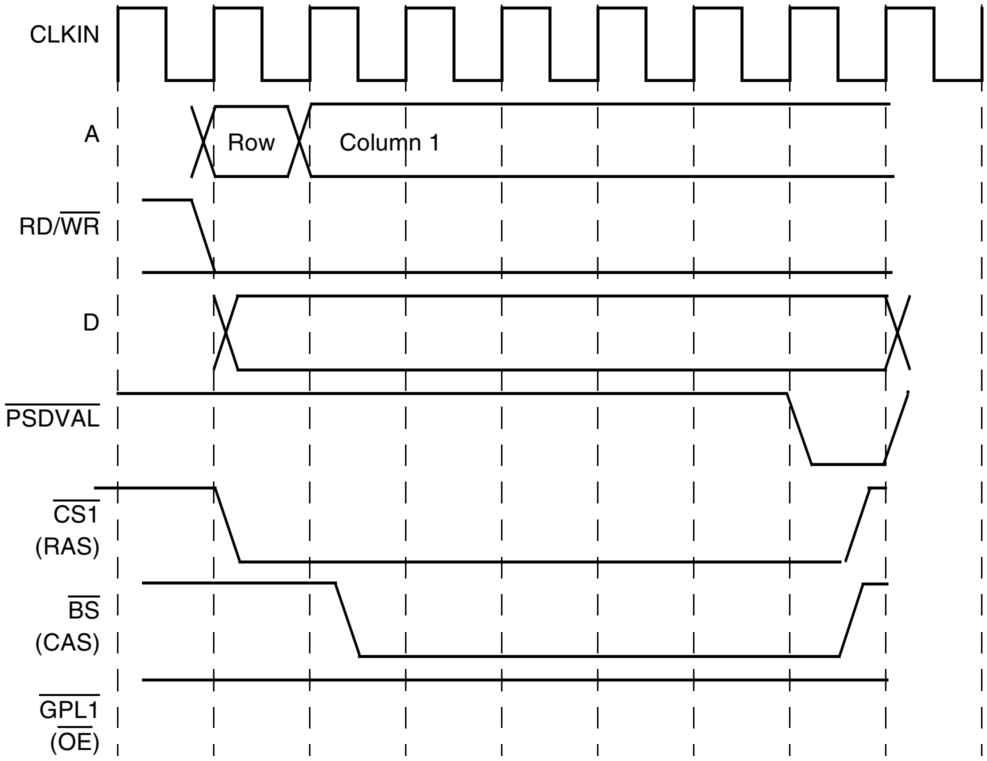
cst1	0	0	0	0	0		Bit 0
cst2	0	0	0	0	0		Bit 1
cst3	0	0	0	0	1		Bit 2
cst4	0	0	0	0	1		Bit 3
bst1	1	1	0	0	0		Bit 4
bst2	1	0	0	0	0		Bit 5
bst3	1	0	0	0	1		Bit 6
bst4	1	0	0	0	1		Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1	0	0	0	0	0		Bit 12
g1t3	0	0	0	0	1		Bit 13
g2t1							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	0	0		Bit 24
exen	0	0	0	0	0		Bit 25
amx0	1	0	0	0	0		Bit 26
amx1	0	0	0	0	0		Bit 27
na	0	0	0	0	0		Bit 28
uta	0	0	0	0	1		Bit 29
todt	0	0	0	0	1		Bit 30
last	0	0	0	0	1		Bit 31
	RSS	**RSS+ 1**	**RSS+ 2**	**RSS+ 3**	**RSS+ 4**		

Figure 11-75. Single-Beat Read Access to EDO DRAM



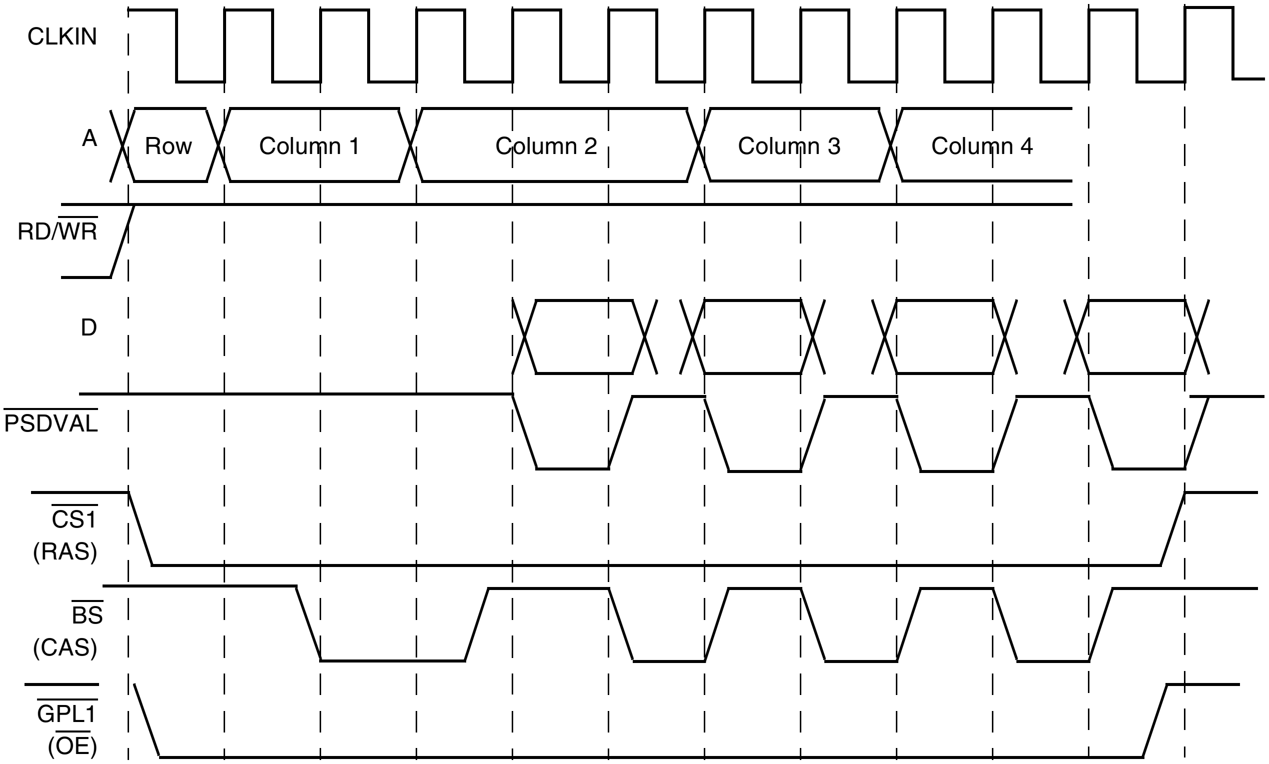
cst1	0	0	0	1		Bit 0
cst2	0	0	0	1		Bit 1
cst3	0	0	1	1		Bit 2
cst4	0	0	1	1		Bit 3
bst1	1	1	0	0		Bit 4
bst2	1	0	0	0		Bit 5
bst3	1	0	0	0		Bit 6
bst4	1	0	0	1		Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1	1	1	1		Bit 12
g1t3	1	1	1	1		Bit 13
g2t1						Bit 14
g2t3						Bit 15
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	0	0	0		Bit 24
exen	0	0	0	0		Bit 25
amx0	1	0	0	0		Bit 26
amx1	0	0	0	0		Bit 27
na	0	0	0	0		Bit 28
uta	0	0	0	1		Bit 29
todt	0	0	0	1		Bit 30
last	0	0	0	1		Bit 31
	WSS	**WSS+ 1**	**WSS+ 2**	**WSS+ 3**		

Figure 11-76. Single-Beat Write Access to EDO DRAM



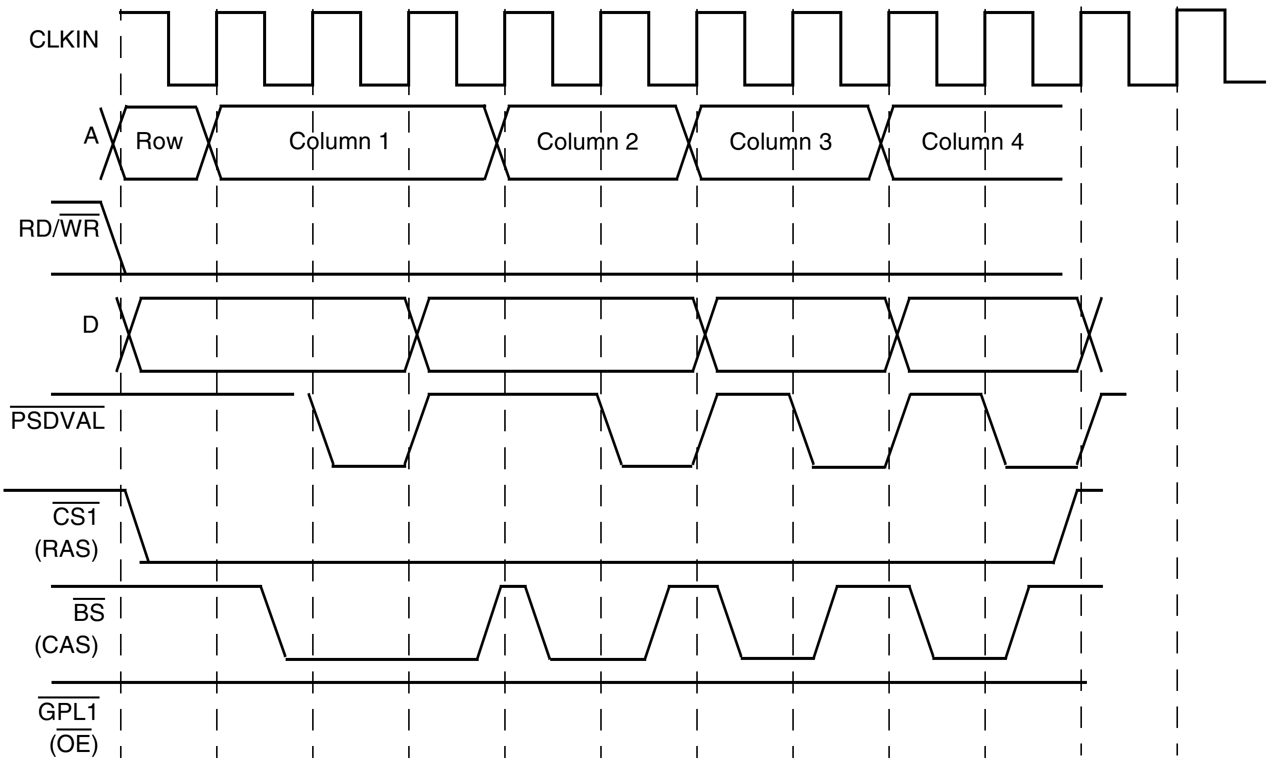
cst1	0	0	0				0		Bit 0
cst2	0	0	0				0		Bit 1
cst3	0	0	0				1		Bit 2
cst4	0	0	0				1		Bit 3
bst1	1	1	0				0		Bit 4
bst2	1	0	0				0		Bit 5
bst3	1	0	0				1		Bit 6
bst4	1	0	0				1		Bit 7
g0l0									Bit 8
g0l1									Bit 9
g0h0									Bit 10
g0h1									Bit 11
g1t1	1	1	1				1		Bit 12
g1t3	1	1	1				1		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]	0	0	1				0		Bit 22
redo[1]	0	0	1				0		Bit 23
loop	0	0	0				0		Bit 24
exen	0	0	0				0		Bit 25
amx0	1	0	0				0		Bit 26
amx1	0	0	0				0		Bit 27
na	0	0	0				0		Bit 28
uta	0	0	0				1		Bit 29
todt	0	0	0				1		Bit 30
last	0	0	0				1		Bit 31
	WSS	**WSS+ 1**	**WSS+ 2**	**REDO 1**	**REDO 2**	**REDO 3**	**WSS+ 3**		

Figure 11-77. Single-Beat Write Access to EDO DRAM Using REDO to Insert Three Wait States



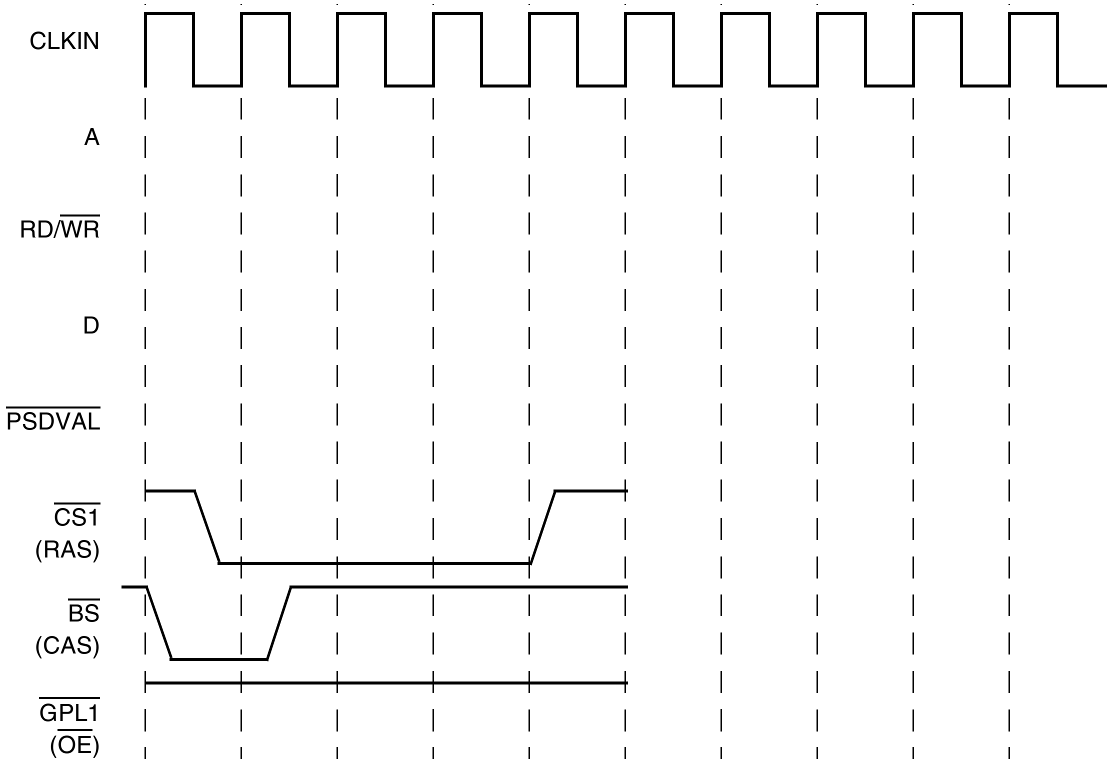
cst1	0	0	0	0	0	0	0	0	0	0	0	Bit 0
cst2	0	0	0	0	0	0	0	0	0	0	0	Bit 1
cst3	0	0	0	0	0	0	0	0	0	0	0	Bit 2
cst4	0	0	0	0	0	0	0	0	0	0	1	Bit 3
bst1	1	1	0	0	1	0	1	0	1	0	1	Bit 4
bst2	1	1	0	0	1	0	1	0	1	0	1	Bit 5
bst3	1	1	0	1	1	0	1	0	1	0	1	Bit 6
bst4	1	0	0	1	1	0	1	0	1	0	1	Bit 7
g0l0												Bit 8
g0l1												Bit 9
g0h0												Bit 10
g0h1												Bit 11
g1t1	0	0	0	0	0	0	0	0	0	0	0	Bit 12
g1t3	0	0	0	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0	0	Bit 24
exen	0	0	0	1	0	1	0	1	0	0	0	Bit 25
amx0	1	0	0	0	0	0	0	0	0	0	0	Bit 26
amx1	0	0	0	0	0	0	0	0	0	0	0	Bit 27
na	0	0	1	0	0	1	0	1	0	0	0	Bit 28
uta	0	0	0	0	1	0	1	0	1	0	1	Bit 29
todt	0	0	0	0	0	0	0	0	0	0	1	Bit 30
last	0	0	0	0	0	0	0	0	0	0	1	Bit 31
	RBS	**RBS+ 1**	**RBS+ 2**	**RBS+ 3**	**RBS+ 4**	**RBS+ 5**	**RBS+ 6**	**RBS+ 7**	**RBS+ 8**	**RBS+ 9**	**RBS+ 10**	

Figure 11-78. Burst Read Access to EDO DRAM



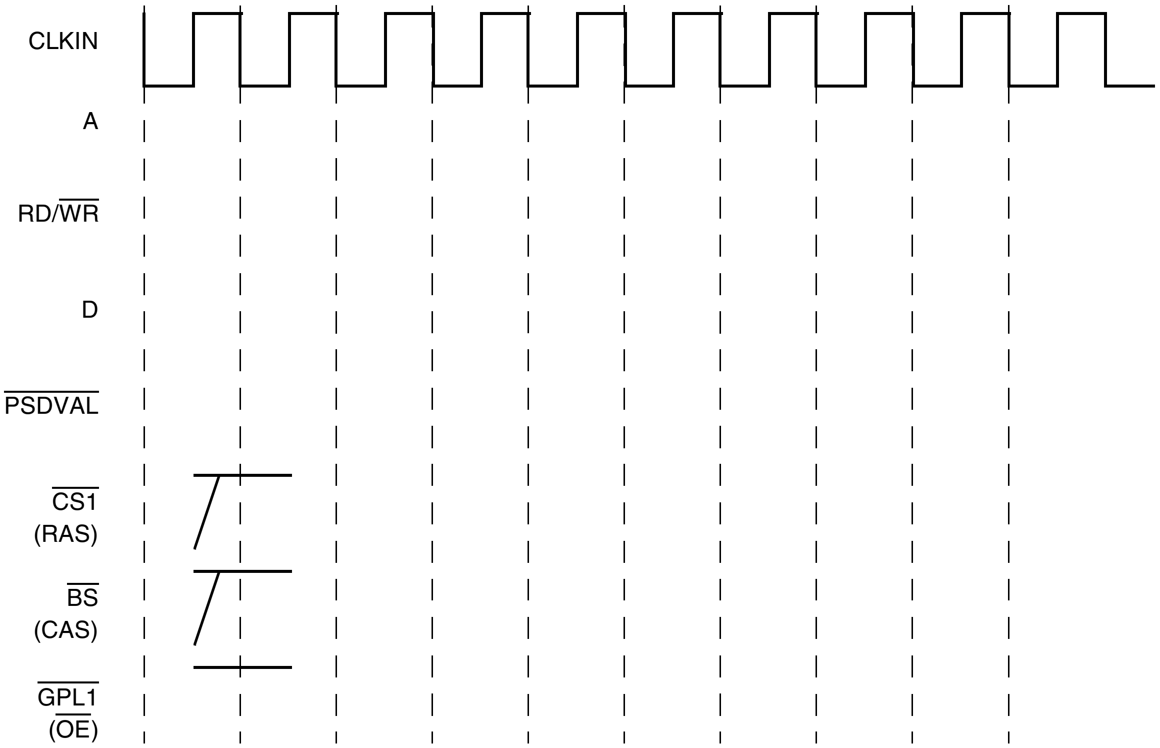
cst1	0	0	0	0	0	0	0	0	0	0		Bit 0
cst2	0	0	0	0	0	0	0	0	0	0		Bit 1
cst3	0	0	0	0	0	0	0	0	0	0		Bit 2
cst4	0	0	0	0	0	0	0	0	0	1		Bit 3
bst1	1	1	0	0	1	0	1	0	1	0		Bit 4
bst2	1	1	0	0	0	0	0	1	0	1		Bit 5
bst3	1	0	0	0	0	1	0	1	0	1		Bit 6
bst4	1	0	0	1	0	1	0	1	0	1		Bit 7
g0l0												Bit 8
g0l1												Bit 9
g0h0												Bit 10
g0h1												Bit 11
g1t1	1	1	1	1	1	1	1	1	1	1		Bit 12
g1t3	1	1	1	1	1	1	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	0	0	0	0	0	0	0	0	0		Bit 24
exen	0	0	0	1	0	1	0	1	0	0		Bit 25
amx0	1	0	0	0	0	0	0	0	0	0		Bit 26
amx1	0	0	0	0	0	0	0	0	0	0		Bit 27
na	0	0	0	1	0	1	0	1	0	0		Bit 28
uta	0	0	1	0	0	1	0	1	0	1		Bit 29
todt	0	0	0	0	0	0	0	0	0	1		Bit 30
last	0	0	0	0	0	0	0	0	0	1		Bit 31
	WBS	**WBS+ 1**	**WBS+ 2**	**WBS+ 3**	**WBS+ 4**	**WBS+ 5**	**WBS+ 6**	**WBS+ 7**	**WBS+ 8**	**WBS+ 9**		

Figure 11-79. Burst Write Access to EDO DRAM



cst1	1	0	0	0	1		Bit 0
cst2	1	0	0	0	1		Bit 1
cst3	0	0	0	0	1		Bit 2
cst4	0	0	0	0	1		Bit 3
bst1	0	0	1	1	1		Bit 4
bst2	0	1	1	1	1		Bit 5
bst3	0	1	1	1	1		Bit 6
bst4	0	1	1	1	1		Bit 7
g0l0							Bit 8
g0l1							Bit 9
g0h0							Bit 10
g0h1							Bit 11
g1t1	1	1	1	1	1		Bit 12
g1t3	1	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	0	0	0	0		Bit 24
exen	0	0	0	0	0		Bit 25
amx0	0	0	0	0	0		Bit 26
amx1	0	0	0	0	0		Bit 27
na	0	0	0	0	0		Bit 28
uta	0	0	0	0	0		Bit 29
todt	0	0	0	0	1		Bit 30
last	0	0	0	0	1		Bit 31
	PTS	**PTS+1**	**PTS+2**	**PTS+3**	**PTS+4**		

Figure 11-80. Refresh Cycle (CBR) to EDO DRAM



cst1 cst2 cst3 cst4	1 1 1 1		Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1			Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		Bit 12 Bit 13
g2t1 g2t3			Bit 14 Bit 15
g3t1 g3t3			Bit 16 Bit 17
g4t1 g4t3			Bit 18 Bit 19
g5t1 g5t3			Bit 20 Bit 21
redo[0] redo[1]			Bit 22 Bit 23
loop exen	0 0		Bit 24 Bit 25
amx0 amx1 na uta todt last	0 0 0 0 1 1		Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	EXS		

Figure 11-81. Exception Cycle For EDO DRAM

11.8 Handling Devices with Slow or Variable Access Times

The memory controller provides two ways to interface with slave devices that are very slow (access time is greater than the maximum allowed by the user programming model) or cannot guarantee a predefined access time (for example some FIFO, hierarchical bus interface, or dual-port memory devices). These mechanisms are as follows:

- The wait mechanism is used only in accesses controlled by the UPM. Setting MxMR[GPLx4DIS] enables this mechanism.
- The external termination ($\overline{\text{GTA}}$) mechanism is used only in accesses controlled by the GPCM. ORx[SETA] specifies whether the access is terminated internally or externally.

The following examples show how the two mechanisms work.

11.8.1 Slow Devices Example

Assume that the core initiates a read cycle from a device whose access time exceeds the maximum allowed by the user programming model.

- The wait solution (UPM)—The core generates a read access from the slow device. The device in turn asserts the wait signal until the data is ready. The core samples data only after the wait signal is negated.
- The external termination solution (GPCM)—The core generates a read access from the slow device, which must generate the asynchronous $\overline{\text{GTA}}$ when it is ready.

11.9 External Master Support (60x-Compatible Mode)

The memory controller supports internal and external bus masters. Accesses from the core or the CPM are considered internal; accesses from an external bus master are external.

External bus master support is available only if the MPC8272 is placed in 60x-compatible mode by setting BCR[EBM]; see Section 4.3.2.1, “Bus Configuration Register (BCR).”

There are two types of external bus masters:

- Any 60x-compatible device with a 64-bit data bus, such as an MPC603e, MPC604e, MPC750, or an MPC2605 (L2 cache) in copy-back mode
- MPC8272

Both of these external bus master types can access a slave MPC8272’s internal registers and dual-port RAM. They can also use the slave’s memory controller to access memory devices on the 60x bus.

11.9.1 60x-Compatible External Masters (non-MPC8272)

A non-MPC8272 external master can perform only 64-bit port accesses when using a slave MPC8272’s memory controller for memory devices assigned to the 60x bus. Also, ECC or RMW-parity are not supported.

For 60x bus compatibility, the following connections should be observed:

- MPC8272’s TSIZ[1:3] should be connected to the external master’s TSIZ[0:2]
- MPC8272’s TSIZ[0] should be pulled down
- MPC8272’s $\overline{\text{PSDVAL}}$ should be pulled up

11.9.2 MPC8272 External Masters

An MPC8272 external master is a 60x-compatible master with additional functionality. As described in the following, it has fewer the restrictions than other 60x-compatible masters:

- Any port size is allowed.
- ECC and RMW-parity are supported.

11.9.3 Extended Controls in 60x-Compatible Mode

In 60x-compatible mode, the memory controller provided extended controls for the glue logic. The extended control consists of the following:

- Memory address latch (ALE) to latch the 60x address for memory use
- The address multiplex pin (GPL5/SDAMUX), which controls external multiplexing for DRAM and SDRAM devices
- LSB address pins (BADDR[27:31]) for incrementing memory addresses
- $\overline{\text{PSDVAL}}$ as a termination to a partial transaction (such as port-size beat access).

11.9.4 Address Incrementing for External Bursting Masters

BADDR[27:31] should be used to generate addresses to memory devices for burst accesses. In 60x-compatible mode, when a master initiates an external bus transaction, it reflects the value of A[27:31] on the first clock cycle of the memory access. These signals are latched by the memory controller and on subsequent clock cycles, BADDR[27:31] increments as programmed in the UPM or after each data beat is sampled in the GPCM or after each READ/WRITE command in the SDRAM machine (the SDRAM machine uses BADDR only for port sizes of 16 or 8 bits).

11.9.5 External Masters Timing

External and internal masters have similar memory access timings. However, because it takes more time to decode the addresses of external masters, memory accesses by external masters start one cycle later than those of internal masters.

As soon as the external master asserts $\overline{\text{TS}}$, the memory controller compares the address with each of its defined valid banks. If a match is found, the memory controller asserts the address latch enable (ALE) and control signals to the memory devices. The memory controller asserts $\overline{\text{PSDVAL}}$ for each data beat to indicate data beat termination on write transactions and data valid on read transactions.

The 60x bus is pipelined. The ALE pins control the external latch that latches the address from the 60x bus and keeps the address stable for the memory access. The memory controller asserts ALE only on the start of new memory controller access.

Figure 11-82 shows the pipelined bus operation in 60x-compatible mode.

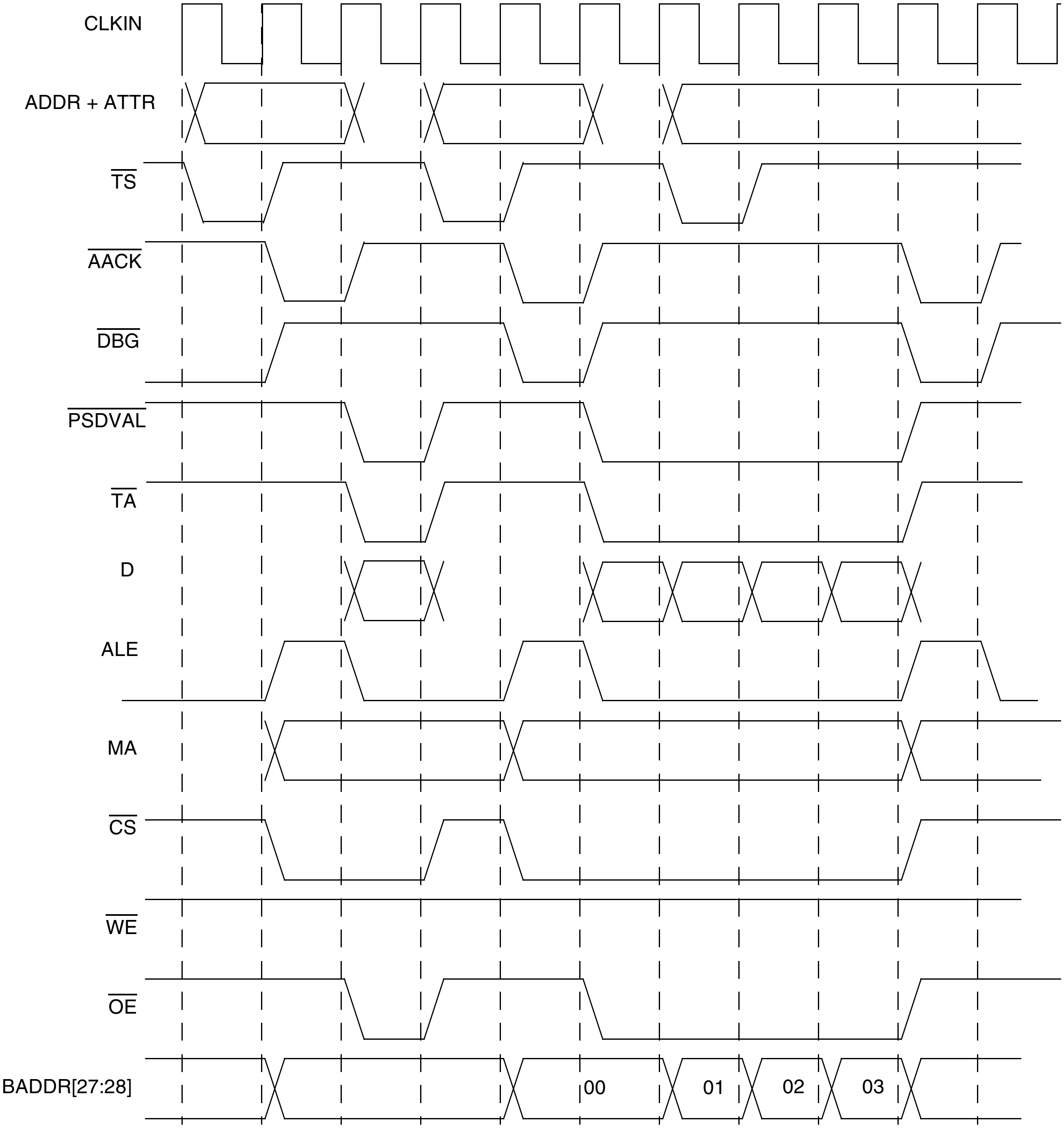


Figure 11-82. Pipelined Bus Operation and Memory Access in 60x-Compatible Mode

Figure 11-83 shows the 1-cycle delay for external master access. For systems that use the 60x bus with low frequency (33 MHz), the 1-cycle delay for external masters can be eliminated by setting BCR[EXDD].

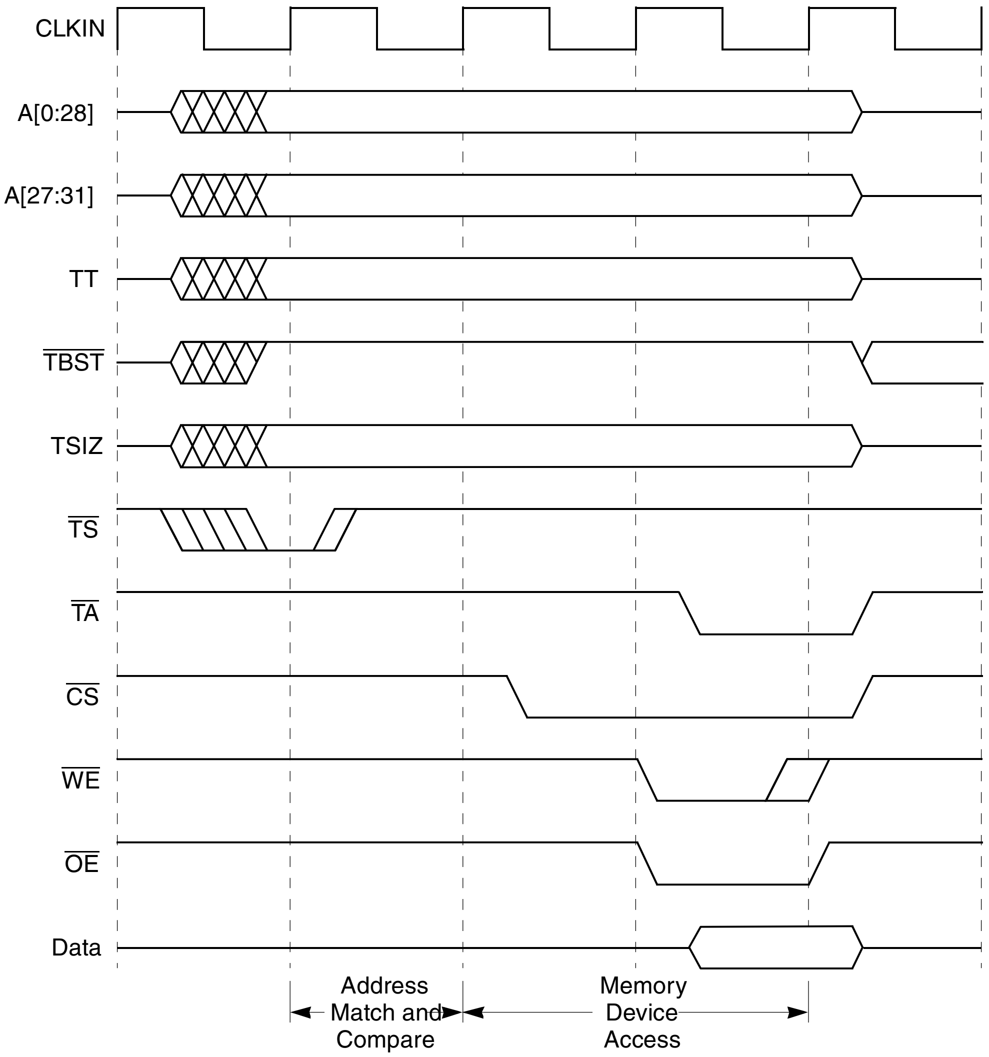


Figure 11-83. External Master Access (GPCM)

11.9.5.1 Example of External Master Using the SDRAM Machine

Figure 11-84 shows an interconnection in which a 60x-compatible external master and the MPC8272 can share access to a SDRAM bank. Note that the address multiplexer is controlled by SDAMUX, while the address latch is controlled by ALE. Also note that because this is a 64-bit port size SDRAM, BADDR is not needed.

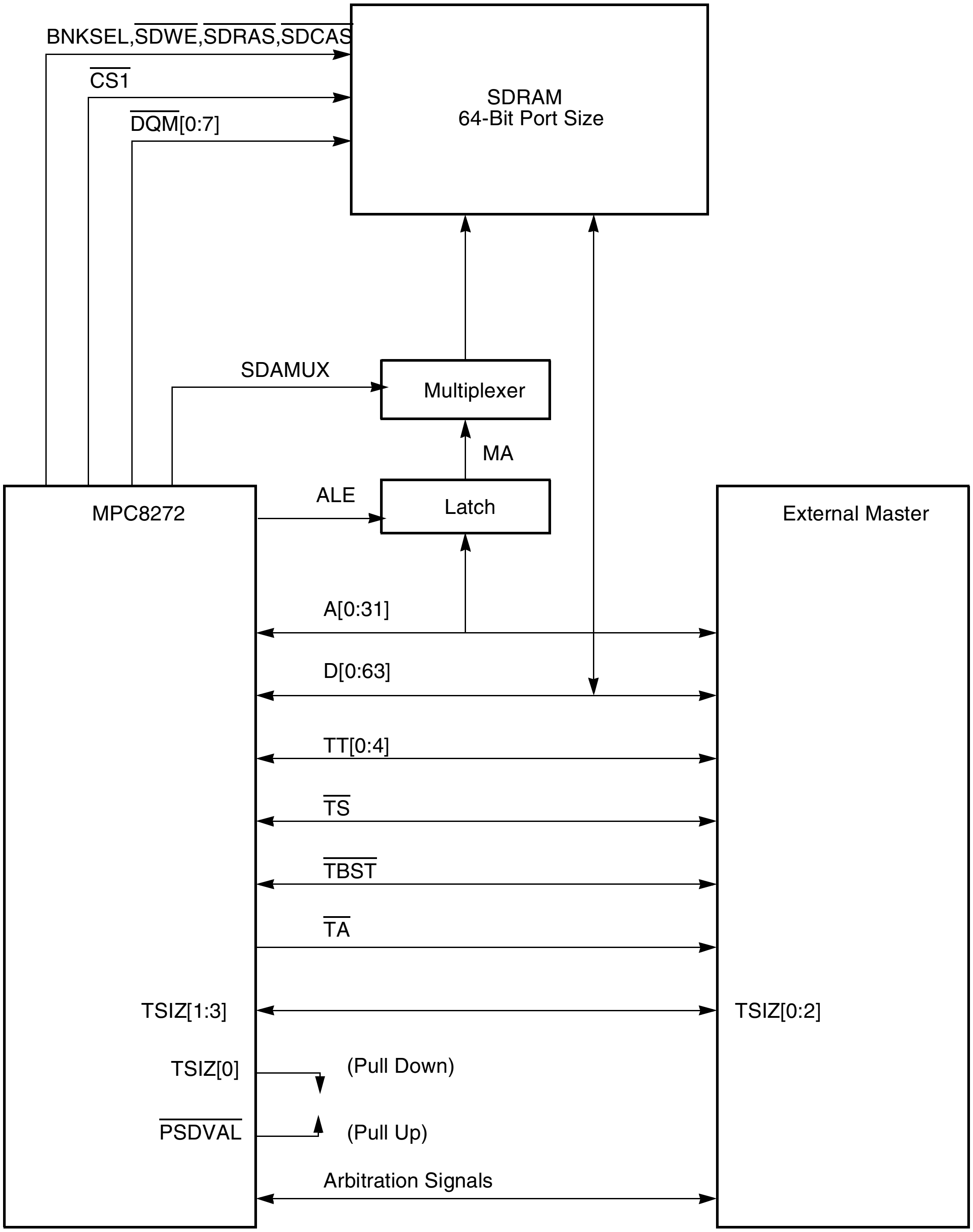


Figure 11-84. External Master Configuration with SDRAM Device

Chapter 12
IEEE 1149.1 Test Access Port

The MPC8272 provides a dedicated user-accessible test access port (TAP) that is fully compatible with the IEEE 1149.1 standard test access port and boundary scan architecture. Problems associated with testing high-density circuit boards have led to the development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The MPC8272's implementation supports circuit-board test strategies based on this standard.

The TAP consists of five dedicated signal pins—a 16-state TAP controller and two test data registers. A boundary scan register links all device signal pins into a single shift register. The test logic, which is implemented using static logic design, is independent of the device system logic. The MPC8272's implementation provides the capability to do the following:

- Perform boundary scan operations to check circuit-board electrical continuity
- Bypass the MPC8272 for a given circuit-board test by effectively reducing the boundary scan register to a single cell
- Disable the output drive to pins during circuit-board testing

NOTE

Precautions must be observed to ensure that the IEEE 1149.1–like test logic does not interfere with nontest operation.

12.1 Overview

The MPC8272's implementation includes a TAP controller, an 8-bit instruction register, and two test registers (a 1-bit bypass register and a 735-bit boundary scan register). Figure 12-1 shows an overview of the MPC8272's scan chain implementation.

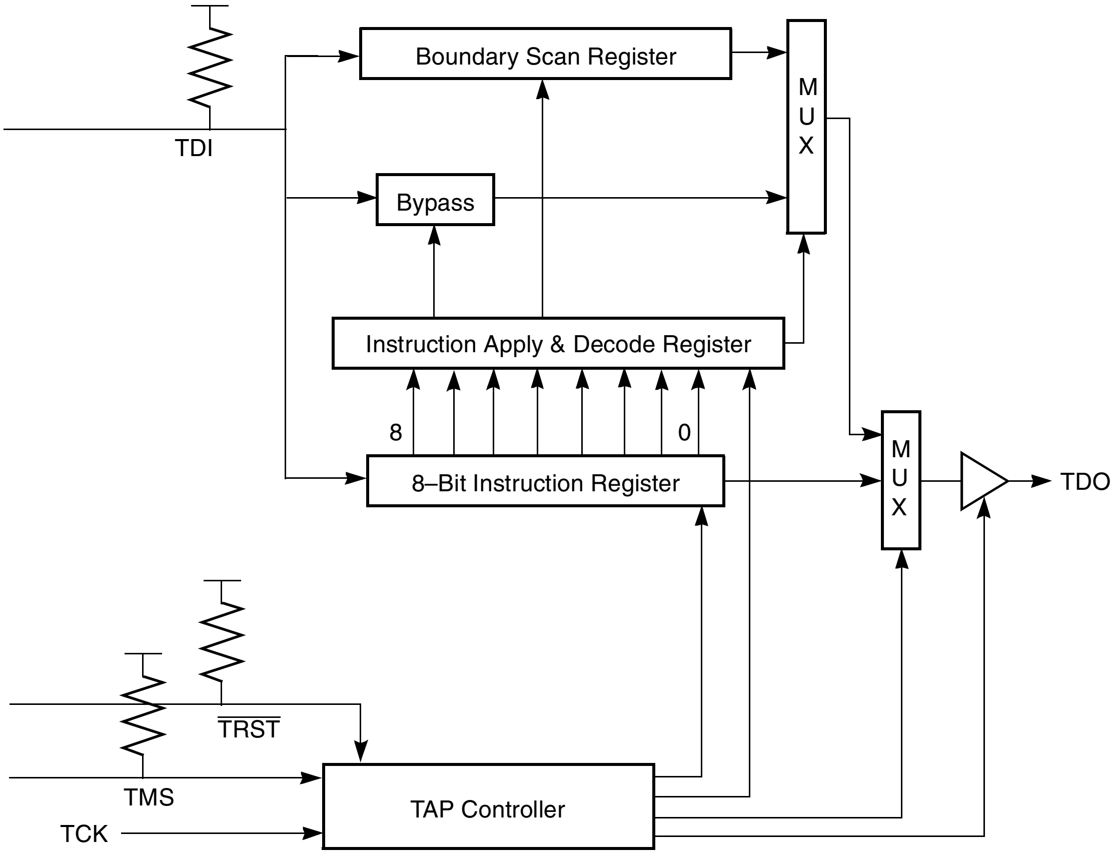


Figure 12-1. Test Logic Block Diagram

The TAP consists of the signals in Table 12-1

Table 12-1. TAP Signals

Signal	Description
TCK	A test clock input to synchronize the test logic
TMS	A test mode select input (with an internal pull-up resistor) that is sampled on the rising edge of TCK to sequence the TAP controller's state machine
TDI	A test data input (with an internal pull-up resistor) that is sampled on the rising edge of TCK
TDO	A data output that can be three-stated and actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK
TRST	An asynchronous reset with an internal pull-up resistor that provides initialization of the TAP controller and other logic required by the standard

12.2 TAP Controller

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The value shown adjacent to each bubble represents the value of the TMS signal sampled on the rising edge of the TCK signal. Figure 12-2 shows the state machine.

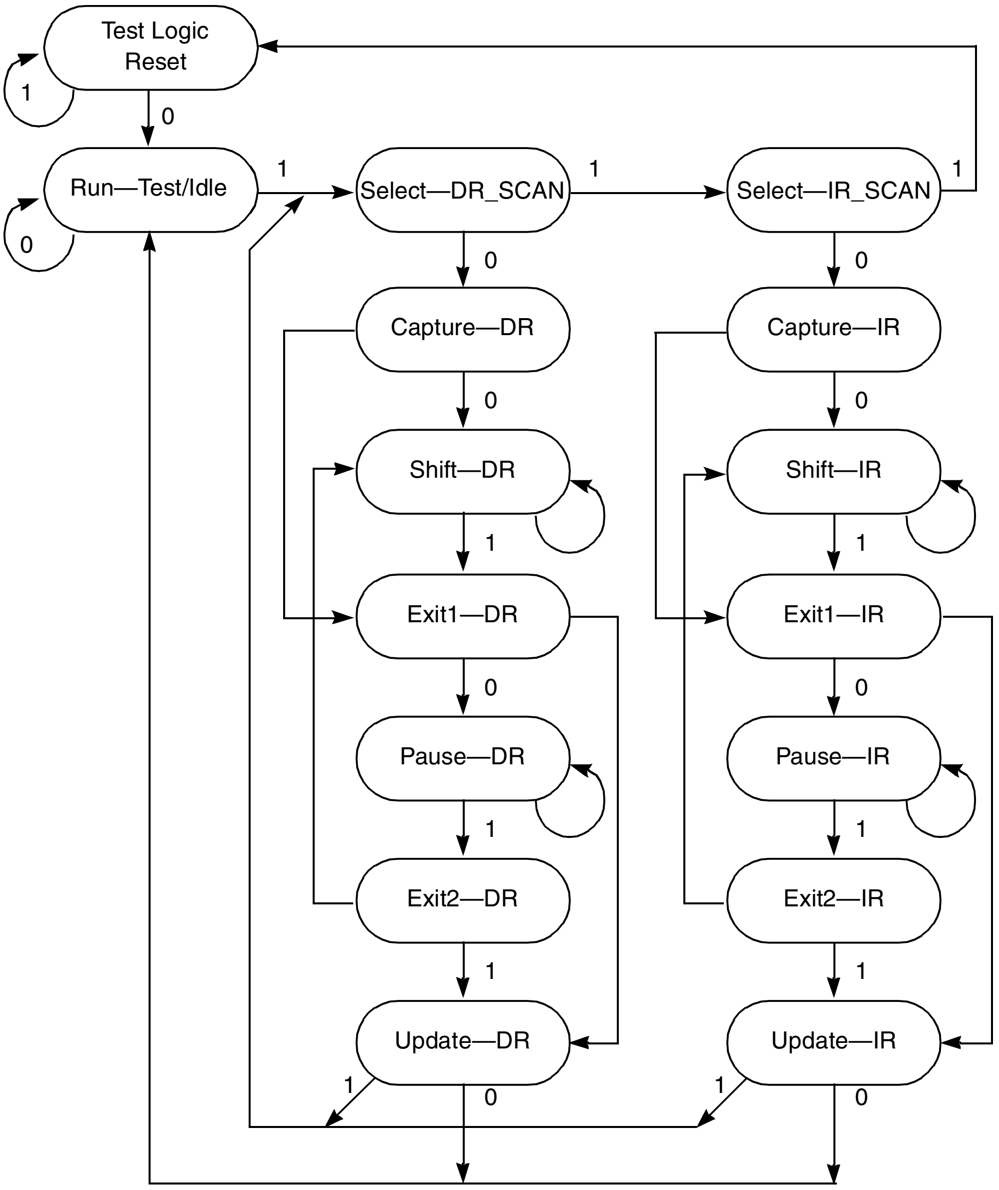


Figure 12-2. TAP Controller State Machine

12.3 Boundary Scan Register

The MPC8272's scan chain implementation has a 735-bit boundary scan register that contains bits for all device signal, clock pins, and associated control signals. The PORESET_B and XFC pins are associated with analog signals and are not included in the boundary scan register. An IEEE 1149.1–compliant boundary-scan register has been included on the MPC8272 that can be connected between TDI and TDO when EXTEST or SAMPLE/PRELOAD instructions are selected. It is used for capturing signal pin data on the input pins, forcing fixed values on the output signal pins, and selecting the direction and drive characteristics (a logic value or high impedance) of the bidirectional and three-state signal pins. Figure 12-3, Figure 12-4, Figure 12-5, and Figure 12-6 show various cell types.

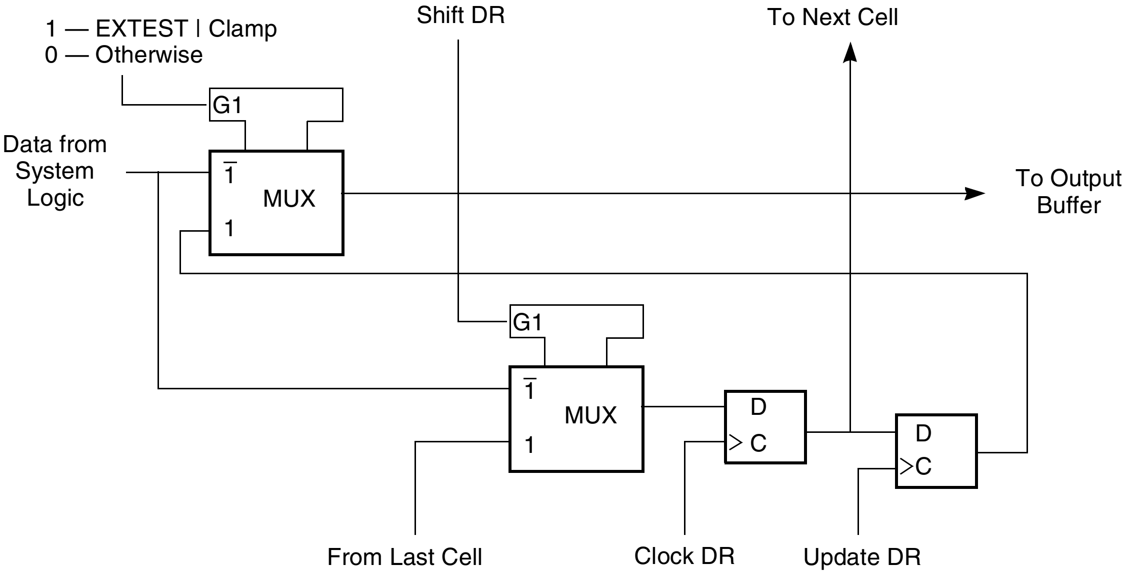


Figure 12-3. Output Pin Cell (O.Pin)

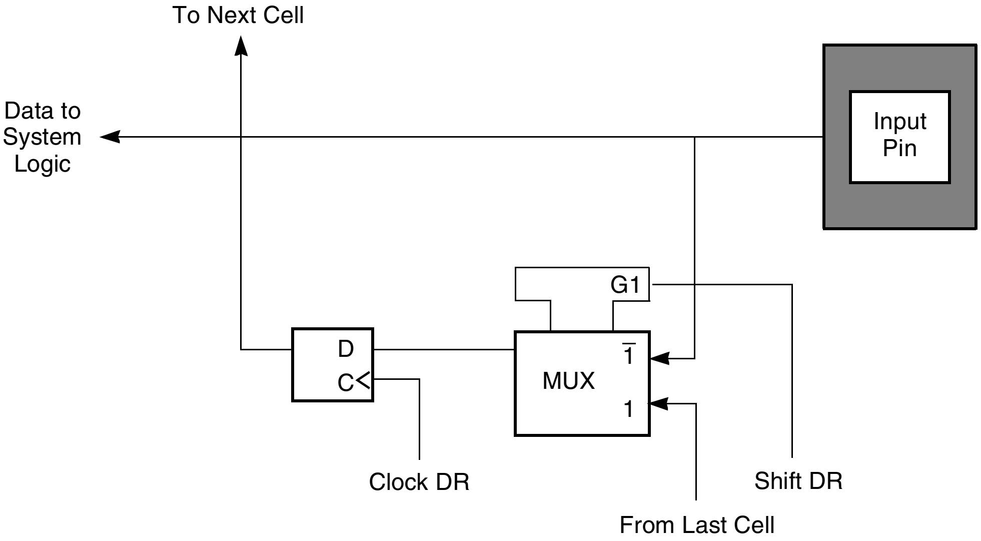


Figure 12-4. Observe-Only Input Pin Cell (I.Obs)

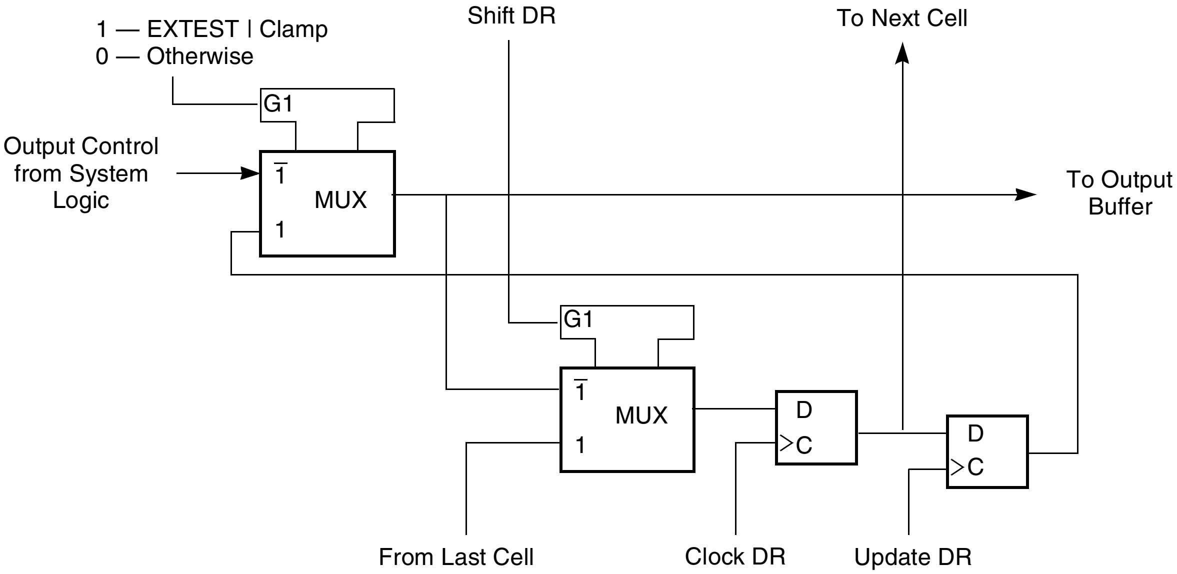


Figure 12-5. Output Control Cell (IO.CTL)

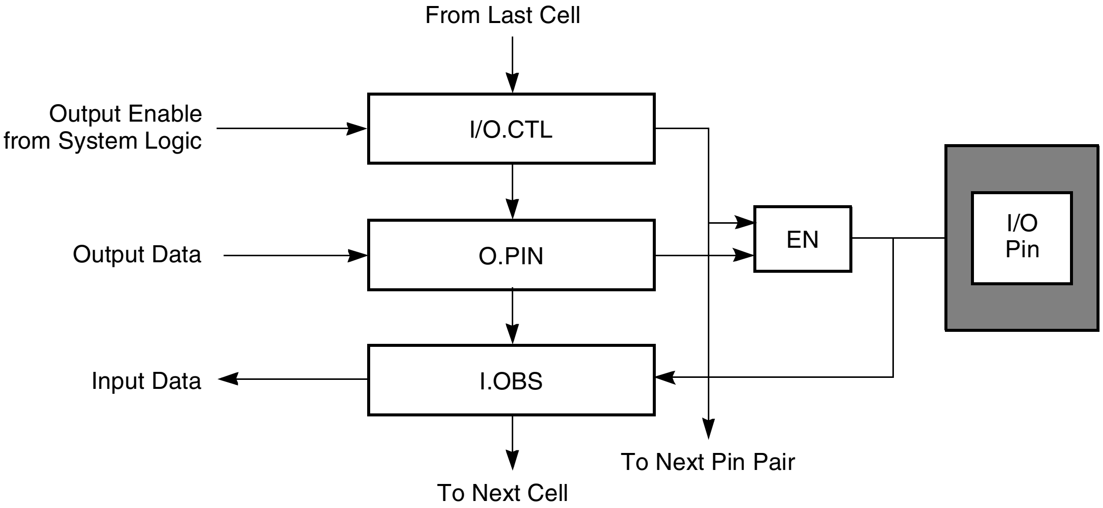


Figure 12-6. General Arrangement of Bidirectional Pin Cells

The control bit value controls the output function of the bidirectional pin. One or more bidirectional data cells can be serially connected to a control cell. Bidirectional pins include two scan cells for data (IO.Cell) as shown in Figure 12-6, and these bits are controlled by the cell shown in Table 12-5.

12.4 Instruction Register

The MPC8272 JTAG implementation includes the public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS) and supports the CLAMP instruction. An additional public instruction, HI-Z, can be used to disable all device output drivers. The MPC8272 includes an 8-bit instruction register (no parity) that consists of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The four bits are used to decode the five unique instructions listed in Table 12-2.

Table 12-2. Instruction Decoding

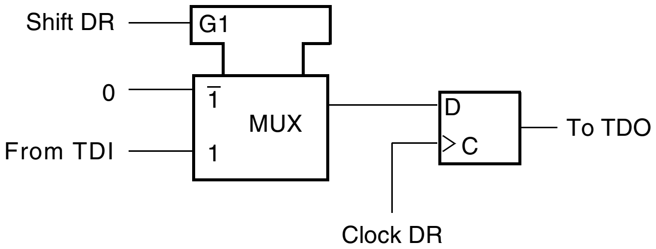
Code[1]								Instruction	Description
B7	**B6**	**B5**	**B4**	**B3**	**B2**	**B1**	**B0**		
0	0	0	0	0	0	0	0	EXTEST	External test. Selects the 475-bit boundary scan register. EXTEST also asserts an internal reset for the MPC8272's system logic to force a known beginning internal state while performing external boundary scan operations. By using the TAP, the register is capable of scanning user-defined values into the output buffers, capturing values presented to input pins, sampling this data into the boundary scan register (device revision B.3 and later), and controlling the output drive of three-state output or bi-directional pins. For more details on the function and use of EXTEST, refer to the IEEE 1149.1 standard.
1	1	0	0	0	0	0	0	SAMPLE/ PRELOAD	Initializes the boundary scan register output cells before the selection of EXTEST. This initialization ensures that known data appears on the outputs when entering an EXTEST instruction. SAMPLE/PRELOAD also provides a chance to obtain a snapshot of system data and control signals. NOTE: Since there is no internal synchronization between the TCK and CLKOUT, the user must provide some form of external synchronization between the JTAG operation at TCK frequency and the system operation CLKOUT frequency to achieve meaningful results.
1	1	1	1	1	1	1	1	BYPASS	The BYPASS instruction creates a shift register path from TDI to the bypass register and, finally, to TDO, circumventing the 475-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the MPC8272 becomes the device under test. It selects the single-bit bypass register as shown below.  Shift DR G1 0 $\overline{1}$ MUX From TDI 1 D C To TDO Clock DR When the bypass register is selected by the current instruction, the shift register stage is cleared on the rising edge of TCK in the capture-DR controller state. Thus, the first bit to be shifted out after selecting the bypass register is always a logic zero.

Table 12-2. Instruction Decoding (continued)

Code[1]								Instruction	Description
B7	B6	B5	B4	B3	B2	B1	B0		
1	1	1	1	0	0	0	0	HI–Z	Provided as a manufacturer’s optional public instruction to avoid back driving the output pins during circuit-board testing. When HI-Z is invoked all output drivers, including the two-state drivers, are turned off (high impedance). The instruction selects the bypass register.
1	1	1	1	0	0	0	1	CLAMP and BYPASS	CLAMP selects the single-bit bypass register as shown in the BYPASS instruction figure above, and the state of all signals driven from the system output pins is completely defined by the data previously shifted into the boundary scan register. For example, using the SAMPLE/PRELOAD instruction.

[1] B0 (lsb) is shifted first.

The parallel output of the instruction register is set to all ones in the test-logic-reset controller state. Notice that this preset state is equivalent to the BYPASS instruction. During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with the CLAMP command code.

12.5 MPC8272 Restrictions

The control afforded by the output enable signals using the boundary-scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the MPC8272’s output drivers are enabled into actively-driven networks.

12.6 Nonscan Chain Operation

In nonscan chain operation, the TCK input does not include an internal pull-up resistor and should be tied high or low to preclude mid-level inputs.

To ensure that the scan chain test logic is kept transparent to the system logic, the TAP controller is forced into the test-logic-reset state. This is done inside the chip by connecting $\overline{\text{TRST}}$ to $\overline{\text{PORESET}}$

TMS should remain logic high, so that the TAP controller does not leave the test-logic-reset state.

Part IV
Communications Processor Module

Intended Audience

This part is intended for system designers who need to implement various communications protocols on the MPC8272. It assumes a basic understanding of the PowerPC exception model and the MPC8272 interrupt structure, as well as a working knowledge of the communications protocols to be used. A complete discussion of these protocols is beyond the scope of this book.

Contents

This part describes the behavior of the MPC8272 communications processor module (CPM) and the RISC communications processor (CP) (note that this is separate from the embedded processor that implements the PowerPC architecture).

It contains the following chapters:

- Chapter 13, “Communications Processor Module Overview,” provides a brief overview of the CPM.
- Chapter 14, “Serial Interface with Time-Slot Assigner,” describes the SIU, which controls system start-up, initialization and operation, protection, as well as the external system bus.
- Chapter 15, “CPM Multiplexing,” describes the CPM multiplexing logic (CMX), which connects the physical layer—UTOPIA, MII, modem lines,
- Chapter 16, “Baud-Rate Generators (BRGs),” describes the eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs, SCCs, and SMCs.
- Chapter 17, “Timers,” describes the timer implementation, which can be configured as four identical 16-bit or two 32-bit general-purpose timers.
- Chapter 18, “SDMA Channels and IDMA Emulation,” describes the two physical serial DMA (SDMA) channels on the MPC8272.
- Chapter 19, “Serial Communications Controllers (SCCs),” describes the four serial communications controllers (SCC), which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 20, “SCC UART Mode,” describes the MPC8272 implementation of the universal asynchronous receiver transmitter (UART) protocol that is used for sending low-speed data between devices.
- Chapter 21, “SCC HDLC Mode,” describes the MPC8272 implementation of HDLC protocol.

- Chapter 22, “SCC BISYNC Mode,” describes the MPC8272 implementation of byte-oriented BISYNC protocol developed by IBM for use in networking products.
- Chapter 23, “SCC Transparent Mode,” describes the MPC8272 implementation of transparent mode (also called totally transparent mode), which provides a clear channel on which the SCC can send or receive serial data without bit-level manipulation.
- Chapter 24, “SCC Ethernet Mode,” describes the MPC8272 implementation of the Ethernet protocol.
- Chapter 25, “SCC AppleTalk Mode,” describes the MPC8272 implementation of AppleTalk.
- Chapter 26, “QMC (QUICC Multi-Channel Controller),” describes the QMC protocol, which emulates up to 64 logical channels within one SCC using the same time-division-multiplexed (TDM) physical interface.
- Chapter 27, “Universal Serial Bus Controller,” describes the MPC8272’s USB controller, including basic operation, the parameter RAM, and registers.
- Chapter 28, “Serial Management Controllers (SMCs),” describes two serial management controllers, full-duplex ports that can be configured independently to support one of three protocols—UART, transparent, or general-circuit interface (GCI).
- Chapter 29, “Fast Communications Controllers (FCCs),” describes the MPC8272’s fast communications controllers (FCCs), which are SCCs optimized for synchronous high-rate protocols.
- Chapter 30, “ATM Controller and AAL0, AAL1, and AAL5 Protocols,” describes the MPC8272 ATM controller, which provides the ATM and AAL layers of the ATM protocol. The ATM controller performs segmentation and reassembly (SAR) functions of AAL5, AAL1, and AAL0, and most of the common parts convergence sublayer (CP-CS) of these protocols.
- Chapter 31, “AAL2 Protocol,” describes he functionality and data structures of ATM adaptation layer type 2 (AAL2) CPS, CPS switching, and SSSAR.
- Chapter 32, “Fast Ethernet Controller,” describes the MPC8272’s implementation of the Ethernet IEEE 802.3 protocol.
- Chapter 33, “FCC HDLC Controller,” describes the FCC implementation of the HDLC protocol.
- Chapter 34, “FCC Transparent Controller,” describes the FCC implementation of the transparent protocol.
- Chapter 35, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface, which allows the MPC8272 to exchange data between other MPC8272 chips, the MC68360, the MC68302, the M68HC11, and M68HC05 microcontroller families, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.
- Chapter 36, “I^{2}C Controller,” describes the MPC8272 implementation of the inter-integrated circuit (I^{2}C®) controller, which allows data to be exchanged with other I^{2}C devices, such as microcontrollers, EEPROMs, real-time clock devices, and A/D converters.
- Chapter 37, “Parallel I/O Ports,” describes the four general-purpose I/O ports A–D. Each signal in the I/O ports can be configured as a general-purpose I/O signal or as a signal dedicated to supporting communications devices, such as SMCs, SCCs, and FCCs.

Suggested Reading

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

MPC82xx Documentation

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.freescale.com. This documentation includes technical specifications, reference materials, and detailed applications notes.

Architecture Documentation

Documentation is available in the following document:

- Programming environments manuals—These books provide information about resources defined by the PowerPC architecture that are common to processors that implement the PowerPC architecture. There are two versions, one that describes the functionality of the combined 32- and 64-bit architecture models and one that describes only the 32-bit model.
  - — *Programming Environments for 32-Bit Implementations of the PowerPC Architecture*, Rev 3(Freescale order #: MPCFPE32B/AD)

For a current list of documentation, refer to http://www.freecale.com.

Conventions

This document uses the following notational conventions:

Bold	Bold entries in figures and tables showing registers and parameter RAM should be initialized by the user.
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.
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 or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In certain contexts, such as in a signal encoding or a bit field, indicates a don’t care.
n	Indicates an undefined numerical value
¬	NOT logical operator

&	AND logical operator
|	OR logical operator

Acronyms and Abbreviations

Table IV-1 contains acronyms and abbreviations used in this document. Note that the meanings for some acronyms (such as SDR1 and DSISR) are historical, and the words for which an acronym stands may not be intuitively obvious.

Table IV-1. Acronyms and Abbreviated Terms

Term	Meaning
AAL	ATM adaptation layer
ABR	Available bit rate
ACR	Allowed cell rate
ALU	Arithmetic logic unit
APC	ATM pace control
ATM	Asynchronous transfer mode
BD	Buffer descriptor
BIST	Built-in self test
BT	Burst tolerance
CBR	Constant bit rate
CEPT	Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations).
C/I	Condition/indication channel used in the GCI protocol
CLP	Cell loss priority
CP	Communications processor
CP-CS	Common part convergence sublayer
CPM	Communications processor module
CPS	Cells per slot
CSMA	Carrier sense multiple access
CSMA/CD	Carrier sense multiple access with collision detection
DMA	Direct memory access
DPLL	Digital phase-locked loop
DPR	Dual-port RAM
DRAM	Dynamic random access memory
DSISR	Register used for determining the source of a DSI exception
EA	Effective address
EEST	Enhanced Ethernet serial transceiver
EPROM	Erasable programmable read-only memory
FBP	Free buffer pool
FIFO	First-in-first-out (buffer)
GCI	General circuit interface

Table IV-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
GCRA	Generic cell rate algorithm (leaky bucket)
GPCM	General-purpose chip-select machine
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
IrDA	Infrared Data Association
ISDN	Integrated services digital network
JTAG	Joint Test Action Group
JTAG	Joint Test Action Group
LAN	Local area network
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
MAC	Multiply accumulate or media access control
MBS	Maximum burst size
MII	Media-independent interface
MSB	Most-significant byte
msb	Most-significant bit
MSR	Machine state register
NaN	Not a number
NIC	Network interface card
NIU	Network interface unit
NMSI	Nonmultiplexed serial interface
NRT	Non-real time
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PDU	Protocol data unit
PCR	Peak cell rate

Table IV-1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
PHY	Physical layer
PPM	Pulse-position modulation
RM	Resource management
RT	Real-time
RTOS	Real-time operating system
Rx	Receive
SAR	Segmentation and reassembly
SCC	Serial communications controller
SCP	Serial control port
SCR	Sustained cell rate
SDLC	Synchronous Data Link Control
SDMA	Serial DMA
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SRAM	Static random access memory
SRTS	Synchronous residual time stamp
TDM	Time-division multiplexed
TE	Terminal endpoint of an ISDN connection
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmit
UBR	Unspecified bit rate
UBR+	Unspecified bit rate with minimum cell rate guarantee
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
USART	Universal synchronous/asynchronous receiver/transmitter
WAN	Wide area network

Chapter 13
Communications Processor Module Overview

The MPC8272's communications processor module (CPM) is a modified version of the MPC8260 PowerQUICC II family's CPM, which supports high bit-rate protocols like ATM and fast Ethernet and supports up to 256 HDLC channels.

13.1 Features

The CPM includes various blocks to provide the system with an efficient way to handle data communication tasks. The following is a list of the CPM's important features:

- Communications processor (CP)
  - One instruction per clock
  - Executes code from internal ROM or dual-port RAM
  - 32-bit RISC architecture
  - Tuned for communication environments: instruction set supports CRC computation and bit manipulation
  - Internal timer
  - Interfaces with the embedded G2_LE core processor through a dual-port RAM and virtual DMA channels for each peripheral controller. (Dual-port RAM size is 16 Kbytes plus 4 Kbytes of dedicated instruction RAM.)
  - Handles serial protocols and virtual DMA
- Two full-duplex fast serial communications controllers (FCCs) support the following protocols:
  - ATM protocol through UTOPIA interface
  - IEEE802.3/Fast Ethernet
  - HDLC
  - Transparent operation
- Three full-duplex serial communications controllers (SCCs) support the following protocols:
  - IEEE 802.3/Ethernet
  - High level/synchronous data link control (HDLC/SDLC)
  - LocalTalk (HDLC-based local area network protocol)
  - Universal asynchronous receiver transmitter (UART)
  - Synchronous UART (1x clock mode)
  - Binary synchronous communication (BISYNC)
  - Totally transparent operation

- Two full-duplex serial management controllers (SMCs) support the following protocols:
  - GCI (ISDN interface) monitor and C/I channels
  - UART
  - Transparent operation
- Serial peripheral interface (SPI) support for master or slave
- I^{2}C bus controller
- Time-slot assigner supports multiplexing of data from any of the SCCs, FCCs, SMCs. The time-slot assigner supports the following TDM formats:
  - T1/CEPT lines
  - T3/E3
  - Pulse code modulation (PCM) highway interface
  - ISDN primary rate
  - Freescale interchip digital link (IDL)
  - General circuit interface (GCI)
  - User-defined interfaces
- Universal serial bus (USB)
- Eight independent baud rate generators (BRGs)
- Four general-purpose 16-bit timers or two 32-bit timers
- General-purpose parallel ports—sixteen parallel I/O lines with interrupt capability

Figure 13-1 shows the MPC8272’s CPM block diagram.

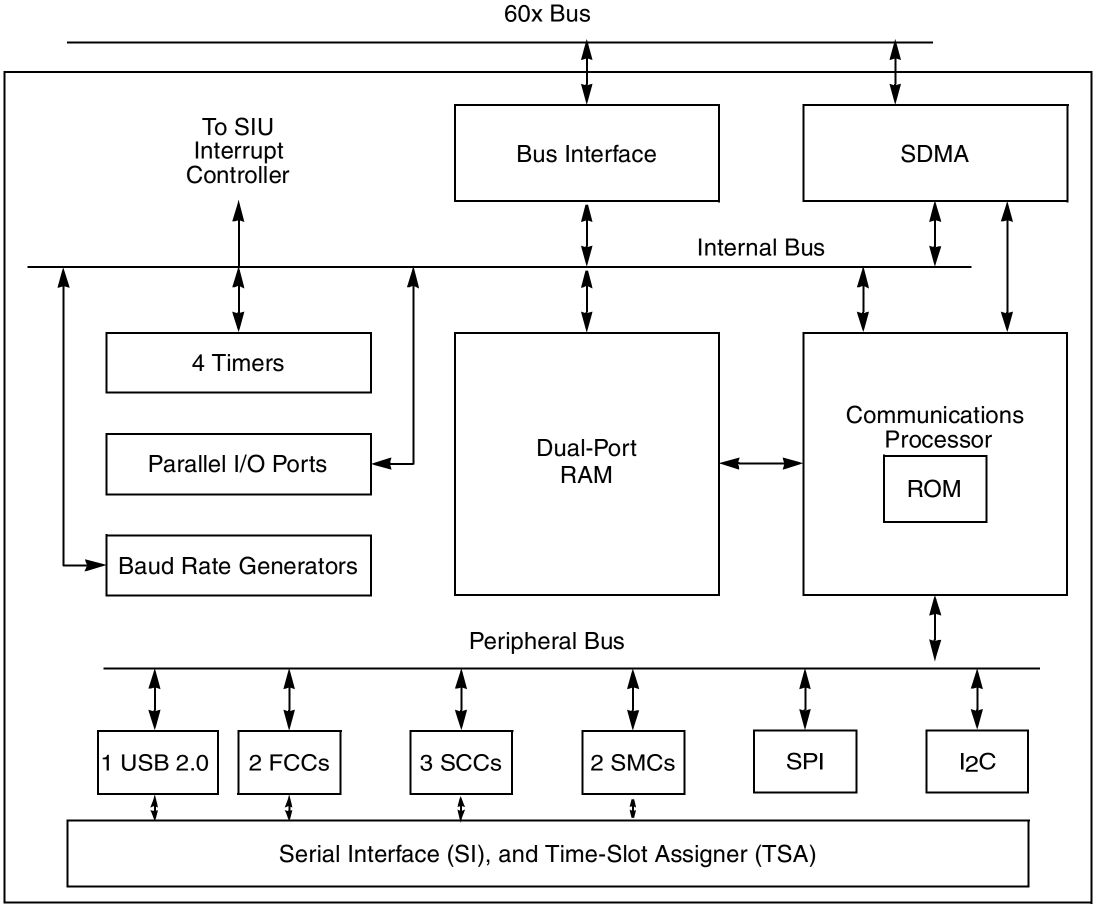


Figure 13-1. MPC8272 CPM Block Diagram

13.2 MPC8272 Serial Configurations

The MPC8272 offers a flexible set of communications capabilities. A subset of the possible configurations using an MPC8272 is shown in Table 13-1.

Table 13-1. Possible MPC8272 Applications

Application	FCC1	FCC2	SCC1	SCC3	SCC4	SMC1	SMC2
ISDN router	FEnet or ATM	FEnet	UART	UART	UART	—	—
ATM switch	ATM	FEnet	UART	—	—	—	—
ATM access	ATM	FEnet	—	—	—	—	—
	ATM	—	UART	—	—	—	—
GSM mobile switching center	FEnet or ATM Backbone	10 M HDLC	—	—	—	—	—

13.3 Communications Processor (CP)

The communications processor (CP), also called the RISC microcontroller, is a 32-bit controller for the CPM that resides on a separate bus from the core and, therefore, can perform tasks independent of the G2_LE core. The CP handles lower-layer communications tasks and DMA control, freeing the core to handle higher-layer activities. The CP works with the peripheral controllers and parallel port to implement user-programmable protocols and manage the serial DMA (SDMA) channels that transfer data between the I/O channels and memory. It also manages the IDMA (independent DMA) channels and contains an internal timer used to implement up to 16 additional software timers.

The CP's architecture and instruction set are optimized for data communications and data processing required by many wire-line and wireless communications standards.

13.3.1 Features

The following is a list of the CP's important features:

- One system clock cycle per instruction
- 32-bit instruction object code
- Executes code from internal ROM or RAM
- 32-bit ALU data path
- 64-bit dual-port RAM access
- Optimized for communications processing
- Performs DMA bursting of serial data from/to dual-port RAM to/from external memory. Note that IDMA cannot burst to dual-port RAM.

13.3.2 CP Block Diagram

The CP contains the following functional units:

- Scheduler and sequencer
- Instruction decoder
- Execution unit
- Load/store unit (LSU)
- Block transfer module (BTM)—moves data between serial FIFO and RAM
- Eight general purpose registers (GPRs)
- Special registers, CRC machine, HDLC framer

The CP also gives SDMA commands to the SDMA. The CP interfaces with the dual-port RAM for loading and storing data and for fetching instructions while running microcode from dual-port RAM.

Figure 13-2 shows the CP block diagram.

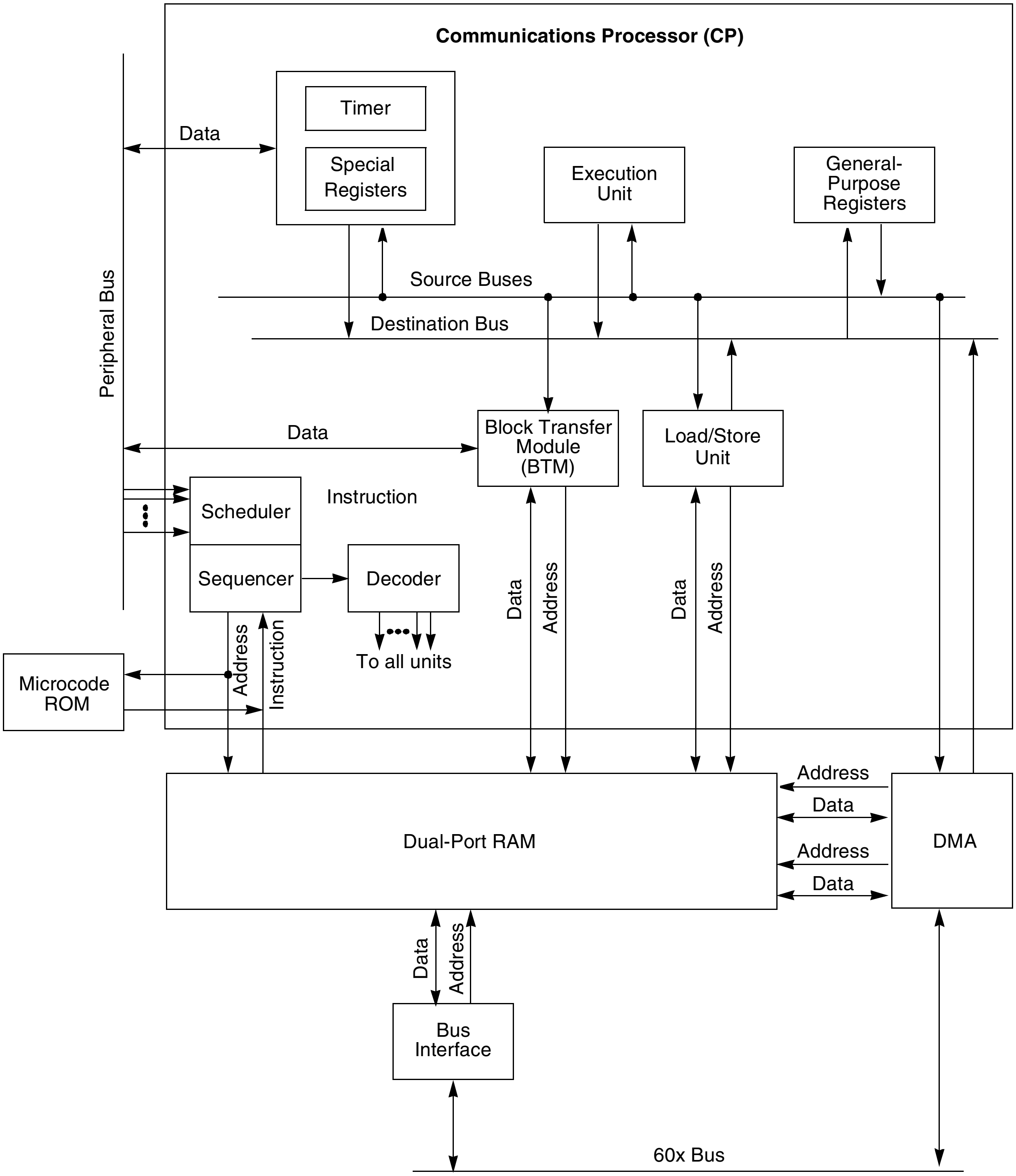


Figure 13-2. Communications Processor (CP) Block Diagram

13.3.3 G2_LE Core Interface

The CP communicates with the G2_LE core in several ways.

- Many parameters are exchanged through the dual-port RAM.
- The CP can execute special commands issued by the core. These commands should only be issued in special situations like exceptions or error recovery.
- The CP generates interrupts through the SIU interrupt controller.
- The G2_LE core can read the CPM status/event registers at any time.

13.3.4 Peripheral Interface

The CP uses the peripheral bus to communicate with all of its peripherals. Each FCC and each SCC has a separate receive and transmit FIFOs. The FCC FIFOs are 192 bytes. The SCC FIFOs are 32 bytes. The SMCs, SPI, and I^{2}C are all double-buffered, creating effective FIFO sizes of two characters.

Table 13-2 shows the order in which the CP handles requests from peripherals from highest to lowest priority.

NOTE

Elevation to emergency status (priority 4) is determined on a per peripheral basis and may depend on a peripheral’s mode of operation. Emergency prioritization among peripherals maintains relative normal prioritization. For example, simultaneous emergency requests from FCC1 transmit and SCC1 transmit is handled in the same order as normal requests (FCC1 transmit).

Table 13-2. Peripheral Prioritization

Priority	Request
1	Reset in the CPCR or $\overline{\text{SRESET}}$
2	SDMA bus error
3	Commands issued to the CPCR
4	Emergency (from FCCs and SCCs)
5	IDMA[2–3] emulation (default—option 1)[1]
6	USB receive
7	USB transmit
8	FCC1 receive
9	FCC1 transmit
10	FCC2 receive
11	FCC2 transmit
12	SCC1 receive
13	SCC1 transmit

Table 13-2. Peripheral Prioritization (continued)

Priority	Request
14	SCC3 receive
15	SCC3 transmit
16	SCC4 receive
17	SCC4 transmit
18	IDMA[2–3] emulation (option 2)[1]
19	SMC1 receive
20	SMC1 transmit
21	SMC2 receive
22	SMC2 transmit
23	SPI receive
24	SPI transmit
25	I^2C receive
26	I^2C transmit
27	RISC timer table
28	IDMA[2–3] emulation (option 3)[1]

[1] The priority of each IDMA channel is programmed independently. See the RCCR[DR*x*QP] description in Section 13.3.6, "RISC Controller Configuration Register (RCCR)."

13.3.5 Execution from RAM

The CP has an option to execute microcode from a dedicated instruction RAM. In this mode, the CP fetches instructions from both the IRAM and its own private ROM. This mode allows Freescale to add new protocols or enhancements to the MPC8272 in the form of RAM microcode packages. If preferred, the user can obtain binary microcode from Freescale and load it into the IRAM.

13.3.6 RISC Controller Configuration Register (RCCR)

The RISC controller configuration register (RCCR), as shown in Figure 13-3, configures the CP to run microcode from ROM or RAM and controls the CP's internal timer.

	0	1	2 … 7	8	9	10 11	12	13	14 15
Field	TIME	—	TIMEP	DR1M	DR2M	DR1QP	EIE	SCD	DR2QP
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x119C4								

	16 … 19	20	21	22	23	24	25	26 27	28	29	30 31
Field	ERAM	EDM1	EDM2	EDM3	EDM4	DR3M	DR4M	DR3QP	DEM12	DEM34	DR4QP
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0X119C6										

Figure 13-3. RISC Controller Configuration Register (RCCR)

RCCR bit fields are described in Table 13-3.

Table 13-3. RISC Controller Configuration Register Field Descriptions

Bits	Name	Description
0	TIME	Timer enable. Enables the CP internal timer that generates a tick to the CP based on the value programmed into the TIMEP field. TIME can be modified at any time to start or stop the scanning of the RISC timer tables.
1	—	Should be cleared
2–7	TIMEP	Timer period. Controls the CP timer tick. The RISC timer tables are scanned on each timer tick and the input to the timer tick generator is the general system clock (133/166 MHz) divided by 1,024. The formula is (TIMEP + 1) × 1,024 = (general system clock period). Thus, a value of 0 stored in these bits gives a timer tick of 1 × (1,024) = 1,024 general system clocks and a value of 63 (decimal) gives a timer tick of 64 × (1,024) = 65,536 general system clocks.
8	DR1M	Must be cleared.
9, 24	DR*2*M, DR*3*M	IDMA*x* request mode. Controls the IDMA request *x* (DREQ*x*) sensitivity mode. DREQ*x* is used to activate IDMA channel *x*. See Section 18.7, “IDMA Interface Signals.” 0 DREQ*x* is edge sensitive (according to EDM*x*). 1 DREQ*x* is level sensitive. **Note:** When DRxM is set to level mode, EDMx determines if IDMA request is active high or active low. Refer to description of RCCR[21–22].
10–11	DR*1*QP	Must be cleared.
12	EIE	External interrupt enable. When EIE is set, CP_INT acts as an external interrupt to the CP. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. 0 CP_INT cannot interrupt the CP. 1 CP_INT will interrupt the CP. **Note:** If EIE = 1, {DR1M, DR4M} must be configured 00 or 01.

Table 13-3. RISC Controller Configuration Register Field Descriptions (continued)

Bits	Name	Description
13	SCD	Scheduler configuration. Configure as instructed in the download process of a Freescale-supplied RAM microcode package. 0 Normal operation 1 Alternate configuration of the scheduler, according to bit 19 (in the ERAM field): If RCCR[19] = 0, the jump table starts at dual-port RAM address 0x0000. If RCCR[19] = 1, the jump table starts at IRAM
14–15, 26–27	DR*2*QP DR*3*QP	IDMA*x* request priority. Controls the priority of DREQ*x* relative to the communications controllers. See Section 18.7, "IDMA Interface Signals." 00 DREQ*x* has more priority than the communications controllers (default). 01 DREQ*x* has less priority than the communications controllers (option 2). 10 DREQ*x* has the lowest priority (option 3). 11 Reserved
16–19	ERAM	Enable RAM microcode. Configure this field as instructed during the downloading process of a Freescale-supplied RAM microcode package. Otherwise, it should not be used. 0000 Disable microcode program execution from the internal RAM. 0100 Microcode is executed from the Instruction RAM. Other combinations of these bits are not valid and must not be used.
20–23	EDM*x*	Edge detect mode. DREQ*x* asserts as follows: 0 Low-to-high change 1 High-to-low change **Note:** When DRxM is set to level mode: 0 DRxM is active high 1 DRxM is active low.
25	DR4M	Should be set.
28	DEM12	Edge detect mode for $\overline{\text{DONE[2]}}$ for IDMA[2]. 0 High-to-low change 1 Low-to-high change
29	DEM34	Edge detect mode for $\overline{\text{DONE[3, 4]}}$ for IDMA[3]. $\overline{\text{DONE[3]}}$ asserts as follows: 0 High-to-low change 1 Low-to-high change
30–31	DR*4*QP	Must be 10.

13.3.7 RISC Time-Stamp Control Register (RTSCR)

The RISC time-stamp control register (RTSCR), shown in Figure 13-4, configures the RISC time-stamp timer (RTSR). The time-stamp timer is used by the ATM and the HDLC controllers. For application examples, see Section 30.5.3, "ABR Flow Control Setup," and Section 33.6, "HDLC Mode Register (FPSMR)."

	0 4	5	6 15
Field	—	RTE	RTPS (Timer Prescale)
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x119DC		

Figure 13-4. RISC Time-Stamp Control Register (RTSCR)

Table 13-4 describes RTSCR fields.

Table 13-4. RTSCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RTE	Time stamp enable. 0 Disable time-stamp timer. 1 Enable time-stamp timer.
6–15	RTPS	Time-stamp timer pre-scale. Must be programmed to generate a 1-μs period input clock to the time-stamp timer. Time-stamp frequency = (CPM frequency)/(RTPS+2)

13.3.8 RISC Time-Stamp Register (RTSR)

The RISC time-stamp register (RTSR), shown in Figure 13-5, contains the time stamp.

	0 15
Field	Time Stamp
Reset	—
R/W	R
Addr	0x119E0

	16 31
Field	Time Stamp
Reset	—
R/W	R
Addr	0X119E2

Figure 13-5. RISC Time-Stamp Register (RTSR)

After reset, setting RTSCR[RTE] causes the time stamp to start counting microseconds from zero.

13.3.9 RISC Microcode Revision Number

The CP writes a revision number stored in its ROM to a dual-port RAM location called REV_NUM that resides in the miscellaneous parameter RAM. The other locations are reserved for future use.

Table 13-5 describes the RISC microcode revision number.

Table 13-5. RISC Microcode Revision Number

Address	Name	Width	Description
RAM Base + 0x8AF0	REV_NUM	Hword	Microcode revision number. 0x00E0.
RAM Base + 0x8AF2	—	Hword	Reserved

13.4 Command Set

The core issues commands to the CP by writing to the CP command register (CPCR). The CPCR rarely needs to be accessed. For example, to terminate the transmission of an SCC’s frame without waiting until the end, a STOP TX command must be issued through the CP command register (CPCR).

13.4.1 CP Command Register (CPCR)

The core should set CPCR[FLG], shown in Figure 13-6, when it issues a command and the CP clears FLG after completing the command, thus indicating to the core that it is ready for the next command. Subsequent commands to the CPCR can be given only after FLG is clear. However, the software reset command issued by setting RST does not depend on the state of FLG, but the core should still set FLG when setting RST.

	0	1 5	6 10	11	15
Field	RST	PAGE	Sub-block code (SBC)	—	FLG
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x119C0				

	16 17	18 25	26 27	28 31
Field	—	MCN	EP[1]	OPCODE
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x119C2			

[1] Only in USB. Otherwise reserved.

Figure 13-6. CP Command Register (CPCR)

Table 13-6 describes CPCR fields.

Table 13-6. CP Command Register Field Descriptions

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>RST</td><td>Software reset command. Set by the core and cleared by the CP. When this command is executed, RST and FLG bit are cleared within two general system clocks. The CPM reset routine is approximately 60 clocks long, but the user can begin initialization of the CPM immediately after this command is issued.
RST is useful when the core wants to reset the registers and parameters for all the channels (FCCs, SCCs, SMCs, SPI, I^{2}C) as well as the CP and RISC timer tables. However, this command does not affect the serial interface (SI2) or parallel I/O registers.</td></tr>
<tr><td>1–5</td><td>PAGE</td><td>Indicates the parameter RAM page number associated with the sub-block being served. See the SBC description for page numbers.</td></tr>
<tr><td>6–10</td><td>SBC</td><td>Sub-block code. Set by the core to specify the sub-block on which the command is to operate. Set according to OPCODE[28-31]. Refer to Table 13-7.
<table>
<tr><th>Sub-block</th><th>Code</th><th>Page</th><th>Sub-block</th><th>Code</th><th>Page</th></tr>
<tr><td>FCC1[1]</td><td>01110: ATM transmit (OPCODE = 1010)
10000: all other commands</td><td>00100</td><td>SPI</td><td>01010</td><td>01001</td></tr>
<tr><td>FCC2[1]</td><td>01110: ATM transmit (OPCODE = 1010)
10001: all other commands</td><td>00101</td><td>I^{2}C</td><td>01011</td><td>01010</td></tr>
<tr><td>SCC1</td><td>00100</td><td>00000</td><td>Timer</td><td>01111</td><td>01010</td></tr>
<tr><td>SCC3</td><td>00110</td><td>00010</td><td>IDMA2</td><td>10101</td><td>01000</td></tr>
<tr><td>SCC4</td><td>00111</td><td>00011</td><td>IDMA3</td><td>10110</td><td>01001</td></tr>
<tr><td>SMC1</td><td>01000</td><td>00111</td><td>USB</td><td>10011</td><td>01011</td></tr>
<tr><td>SMC2</td><td>01001</td><td>01000</td><td></td><td></td><td></td></tr>
<tr><td>RAND</td><td>01110</td><td>01010</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>11–14</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>15</td><td>FLG</td><td>Command semaphore flag. Set by the core and cleared by the CP.
0 The CP is ready to receive a new command.
1 The CPCR contains a command that the CP is currently processing. The CP clears this bit at the end of command execution or after reset.</td></tr>
<tr><td>16–17</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>18-25</td><td>MCN</td><td>MCN: In FCC protocols, this field contains the protocol code as follows
0x00 HDLC, Transparent
0x0A ATM
0x0C Ethernet</td></tr>
</table>

Table 13-6. CP Command Register Field Descriptions (continued)

Bit	Name	Description
26–27	EP	Endpoint. Logical pipe number (only in USB) 00 ENDPOINT 0 01 ENDPOINT 1 10 ENDPOINT 2 11 ENDPOINT 3
28–31	OPCODE	Operation code. Settings are listed in Table 13-7.

[1] Set according to OPCODE[28–31]. If OPCODE is 1010, SBC must be 01110. Refer to Table 13-7. ATM functionality is not available on the MPC8248 and MPC8247.

13.4.1.1 CP Commands

The CP command opcodes are shown in Table 13-7.

Table 13-7. CP Command Opcodes

Opcode	Channel									
	FCC	USB	SCC	SMC (UART/ Transparent)	SMC (GCI)	SPI	I^2C	IDMA	Timer	Special
0000	INIT RX AND TX PARAMS	—	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	—	—	—
0001	INIT RX PARAMS	—	INIT RX PARAMS	INIT RX PARAMS	—	INIT RX PARAMS	INIT RX PARAMS	—	—	—
0010	INIT TX PARAMS	—	INIT TX PARAMS	INIT TX PARAMS	—	INIT TX PARAMS	INIT TX PARAMS	—	—	—
0011	ENTER HUNT MODE	—	ENTER HUNT MODE	ENTER HUNT MODE	—	—	—	—	—	—
0100	STOP TX	—	STOP TX	STOP TX	—	—	—	—	—	—
0101	GRACEFUL STOP TX	—	GRACEFUL STOP TX	—	—	—	—	—	—	—
0110	RESTART TX	—	RESTART TX	RESTART TX	—	—	—	—	—	—
0111	—	—	CLOSE RX BD	CLOSE RX BD	—	CLOSE RX BD	CLOSE RX BD	—	—	—
1000	SET GROUP ADDRESS	—	SET GROUP ADDRESS	—	—	—	—	—	SET TIMER	—
1001	—	—	—	—	GCI TIMEOUT	—	—	START IDMA	—	—
1010	ATM TRANSMIT COMMAND[1]	USB STOP TX ENDPOINT	RESET BCS	—	GCI ABORT REQUEST	—	—	—	—	—

Table 13-7. CP Command Opcodes (continued)

Opcode	Channel									
	FCC	USB	SCC	SMC (UART/ Transparent)	SMC (GCI)	SPI	I^2C	IDMA	Timer	Special
1011	—	USB RESTART TX ENDPOINT	—	—	—	—	—	STOP IDMA	—	—
1100	—	—	QMC STOP TRANSMIT	—	—	—	—	—	—	—
1101	—	—	QMC STOP RECEIVE	—	—	—	—	—	—	—
11xx	Undefined. Reserved for use by Freescale-supplied RAM microcodes.									

[1] See FCC1 and FCC2 in SBC[6–10] in Table 13-6. Not available on the MPC8248 and the MPC8247.

NOTE

If a reserved command is issued, the CPM enters an unknown state that requires an external reset to recover.

The commands in Table 13-7 are described in Table 13-8.

Table 13-8. Command Descriptions

Command	Description
INIT TX AND RX PARAMS	Initialize transmit and receive parameters. Initializes the transmit and receive parameters in the parameter RAM to the values that they had after the last reset of the CP. This command is especially useful when switching protocols on a given peripheral controller.
INIT RX PARAMS	Initialize receive parameters. Initializes the receive parameters of the peripheral controller.
INIT TX PARAMS	Initialize transmit parameters. Initializes the transmit parameters of the peripheral controller.
ENTER HUNT MODE	Enter hunt mode. Causes the receiver to stop receiving and begin looking for a new frame. The exact operation of this command may vary depending on the protocol used.
STOP TX	Stop transmission. Aborts the transmission from this channel as soon as the transmit FIFO has been emptied. It should be used in cases where transmission needs to be stopped as quickly as possible. Transmission proceeds when the RESTART command is issued.
GRACEFUL STOP TX	Graceful stop transmission. Stops the transmission from this channel as soon as the current frame has been fully transmitted from the transmit FIFO. Transmission proceeds when the RESTART command is issued and the R-bit is set in the next TxBD.
RESTART TX	Restart transmission. Once the STOP TX command has been issued, this command is used to restart transmission at the current BD.
USB STOP TX ENDPOINT	See Section 27.7, “USB CP Commands.”
USB RESTART TX ENDPOINT	See Section 27.7, “USB CP Commands.”

Table 13-8. Command Descriptions (continued)

Command	Description
CLOSE RXBD	Close RxBD. Causes the receiver to close the current RxBD, making the receive buffer immediately available for manipulation by the user. Reception continues using the next available BD. Can be used to access the buffer without waiting until the buffer is completely filled by the SCC.
START IDMA	See Section 18.9, "IDMA Commands."
STOP IDMA	See Section 18.9, "IDMA Commands."
SET TIMER	Set timer. Activates, deactivates, or reconfigures one of the 16 timers in the RISC timer table.
SET GROUP ADDRESS	Set group address. Sets a bit in the hash table for the Ethernet logical group address recognition function.
GCI ABORT REQUEST	GCI abort request. The GCI receiver sends an abort request on the E-bit.
GCI TIMEOUT	GCI time-out. The GCI performs the timeout function.
RESET BCS	Reset block check sequence. Used in BISYNC mode to reset the block check sequence calculation.
ATM TRANSMIT[1]	See Section 30.15, "ATM Transmit Command."
RANDOM NUMBER	Generate a random number and put it in dual-port RAM; see RAND in Table 13-10.

[1] Not available on the MPC8248 and the MPC8247.

13.4.2 Command Register Example

To perform a complete reset of the CP, the value 0x8001_0000 should be written to the CPCR. Following this command, the CPCR returns the value 0x0000_0000 after 2 clocks.

13.4.3 Command Execution Latency

The worst-case command execution latency is 200 clocks and the typical command execution latency is about 40 clocks.

13.5 Dual-Port RAM

The CPM static RAM (16 Kbyte) devices is shown in Figure 13-7.

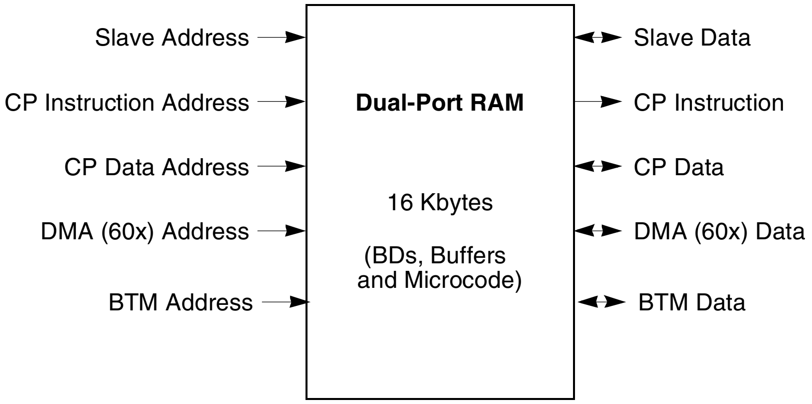


Figure 13-7. Dual-Port RAM Block Diagram

The dual-port RAM can be accessed by the following:

- CP load/store unit
- CP block transfer module (BTM)
- CP instruction fetcher (when executing microcode from RAM)
- G2_LE 60x slave
- SDMA 60x bus

Note that the dual-port RAM should not be cached.

Figure 13-8 shows the memory map of the dual-port RAM.

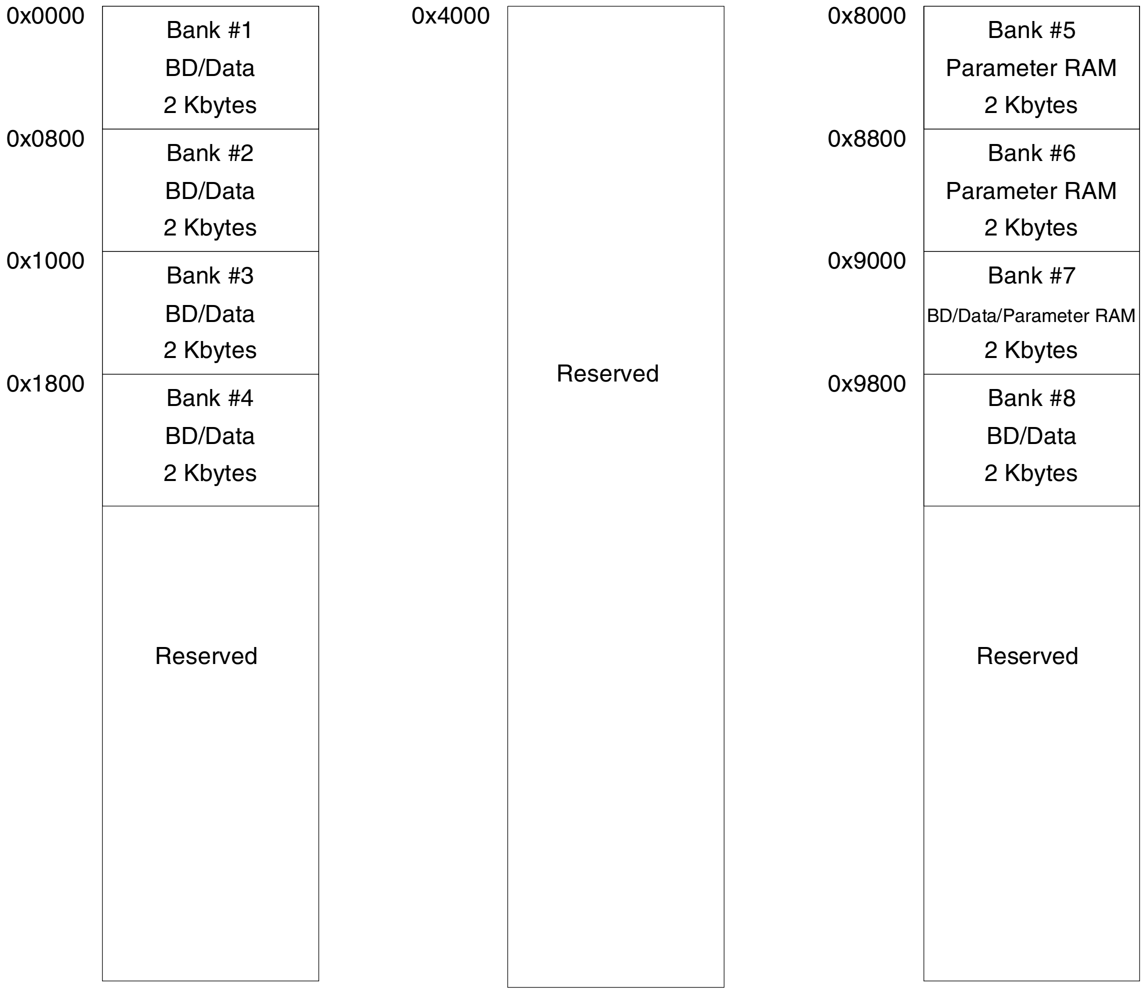


Figure 13-8. Dual-Port RAM Memory Map

The dual-port RAM data bus is 64-bits wide. The RAM is used for the following tasks:

- To store parameters associated with the FCCs, SCCs, SMCs, SPI, I^{2}C, USB, and IDMAs in the parameter RAM
- To store buffer descriptors (BDs)
- To hold data buffers (optional because data can also be stored in external memory)

- For temporary storage of FCC data moving to/from an FCC FIFO (using the BTM) from/to external memory (using SDMA)
- For additional scratch-pad RAM space for user software

The RAM is designed to serve multiple requests at the same cycle, as long as they are not in the same bank.

Only the parameters in the parameter RAM and the microcode RAM option require fixed addresses to be used. The BDs, buffer data, and scratchpad RAM can be located in the dual-port system RAM or in any unused parameter RAM, for example, in the area made available when a peripheral controller or sub-block is not being used.

13.5.1 Buffer Descriptors (BDs)

The peripheral controllers (FCCs, SCCs, SMCs, SPI, and I^{2}C) always use BDs for controlling buffers; their BD formats are all the same, as shown in Table 13-9.

Table 13-9. Buffer Descriptor Format

Address	Descriptor
Offset + 0	Status and control
Offset + 2	Data length
Offset + 4	High-order buffer pointer
Offset + 6	Low-order buffer pointer

If the IDMA is used in the buffer chaining or auto-buffer mode, the IDMA channel also uses BDs. They are described in Section 18.3, “IDMA Emulation.”

NOTE

The CPM accesses BDs by initiating a DMA cycle on the 60x bus. If BDs are located in DPRAM, the CPM initiates a cycle on the 60x bus because DPRAM is a slave device on the 60x bus. Therefore, a system design should not plan to access the 60x bus simultaneously with DPRAM BD fetches.

13.5.2 Parameter RAM

The CPM maintains a section of RAM called the parameter RAM, that contains many parameters for the operation of the FCCs, SCCs, SMCs, SPI, I^{2}C, USB, and IDMA channels. An overview of the parameter RAM structure is shown in Table 13-10.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM. For example, the Ethernet parameter RAM is defined differently in some locations from the HDLC-specific parameter RAM.

Table 13-10. Parameter RAM

Page	Address [1]	Peripheral	Size (Bytes)
1	0x8000	SCC1	256
2	0x8100	Reserved	256
3	0x8200	SCC3	256
4	0x8300	SCC4	256
5	0x8400	FCC1	256
6	0x8500	FCC2	256
7	0x8600	Reserved	256
8	0x8700	Reserved	128
	0x8780	Reserved	124
	0x87FC	SMC1_BASE	2
	0x87FE	Reserved	2
9	0x8800	Reserved	128
	0x8880	Reserved	124
	0x88FC	SMC2_BASE	2
	0x88FE	IDMA2_BASE	2
10	0x8900	Reserved	252
	0x89FC	SPI_BASE	2
	0x89FE	IDMA3_BASE	2
11	0x8A00	Reserved	224
	0x8AE0	RISC timers	16
	0x8AF0	REV_NUM[2]	2
	0x8AF2	Reserved	2
	0x8AF4	Reserved	4
	0x8AF8	RAND	4
	0x8AFC	I^{2}C_BASE	2
	0x8AFE	Reserved	2
12	0x8B00	USB	256
13-16	0x8C00	Reserved	1024

[1] Offset from RAM_BASE.

[2] Refer to Table 13-5.

13.6 RISC Timer Tables

The CP can control up to 16 software timers that are separate from the 4 general-purpose timers and the BRGs in the CPM. These timers are best used in protocols that do not require extreme precision, but in which it is preferable to free the core from scanning the software's timer tables. These timers are clocked from an internal timer that only the CP uses. The following is a list of the 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 7.7 µs at 133 MHz (6.17 µs at 166 MHz)
- Maximum timeout period of 31.8 seconds at 133 MHz (25.5 seconds at 166 MHz)
- Continuously updated reference counter

All operations on the RISC timer tables are based on a fundamental tick of the CP's internal timer that is programmed in the RCCR. The tick is a multiple of 1,024 general system clocks; see Section 13.3.6, "RISC Controller Configuration Register (RCCR)."

The RISC timer tables have the lowest priority of all CP operations. Therefore, if the CP is so busy with other tasks that it does not have time to service the timer during a tick interval, one or more timer may not be updated accurately. This behavior can be used to estimate the worst-case loading of the CP; see Section 13.6.10, "Using the RISC Timers to Track CP Loading."

The timer table is configured using the RCCR, the timer table parameter RAM, and the RISC controller timer event/mask registers (RTER/RTMR), and by issuing SET TIMER to the CPCR.

13.6.1 RISC Timer Table Parameter RAM

Two areas of dual-port RAM, shown in Figure 13-9, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

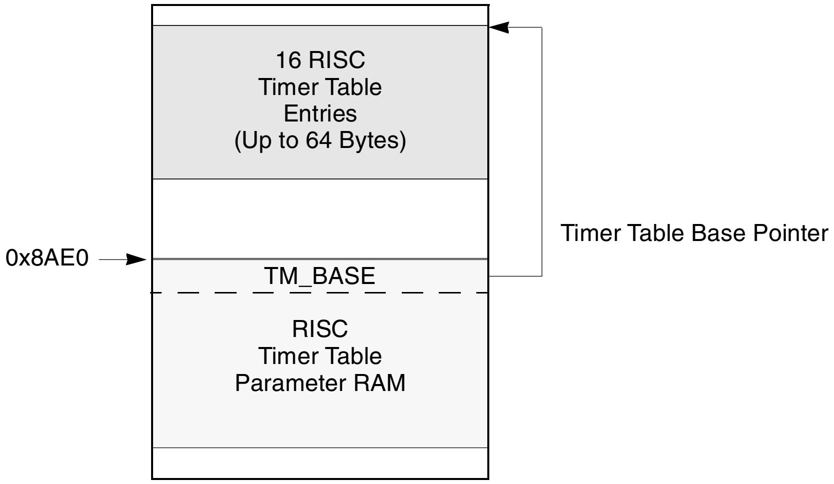


Figure 13-9. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 13-11.

Table 13-11. RISC Timer Table Parameter RAM

Offset[1]	Name	Description
0x00	**TM_BASE**	RISC timer table base address. The actual timers are a small block of memory in the dual-port RAM. TM_BASE is the offset from the beginning of the dual-port RAM where that block resides. Four bytes must be reserved at the TM_BASE for each timer used, (64 bytes if all 16 timers are used). If fewer than 16 timers are used, timers should be allocated in ascending order to save space. For example, only 8 bytes are required if two timers are needed and RISC timers 0 and 1 are enabled. TM_BASE should be word-aligned.
0x02	TM_PTR	RISC timer table pointer. This value is used exclusively by the CP to point to the next timer accessed in the timer table. It should not be modified by the user.
0x04	R_TMR	RISC timer mode register. This value is used exclusively by the CP to store the mode of the timer—one-shot (bit is 0) or restart (bit is 1). R_TMR should not be modified by the user. The SET TIMER command should be used instead.
0x06	R_TMV	RISC timer valid register. Used exclusively by the CP to determine if a timer is currently enabled. If the corresponding timer is enabled, a bit is 1. R_TMV should not be modified by the user. The SET TIMER command should be used instead.
0x08	**TM_CMD**	RISC timer command register. Used as a parameter location when the SET TIMER command is issued. The user should write this location before issuing the SET TIMER command. This register is defined in Section 13.6.2, “RISC Timer Command Register (TM_CMD).”
0x0C	**TM_CNT**	RISC timer internal count. A tick counter that the CP updates after each tick. The update occurs after the CP complete scanning the timer table.All 16 timers are scanned every tick interval regardless of whether any of them is enabled.It is updated if the CP’s internal timer is enabled, regardless of whether any of the 16 timers are enabled and it can be used to track the number of ticks the CP receives and responds to.TM_CNT is updated only after the last timer (timer 15) has been serviced. If the CP is so busy with other tasks that it does not have time to service all the timers during a tick interval, and timer 15 has not been serviced, then TM_CNT would not be updated in that tick interval.

[1] Offset from timer base address (0x8AE0).

13.6.2 RISC Timer Command Register (TM_CMD)

Figure 13-10 shows the RISC timer command register (TM_CMD).

	0	1	2–11	12–15
Field	**V**	**R**	—	**TN**

	16–31
Field	**TIMER PERIOD (TP)**

Figure 13-10. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 13-12.

Table 13-12. TM_CMD Field Descriptions

Bits	Name	Description
0	**V**	Valid. This bit should be set to enable the timer and cleared to disable it.
1	**R**	Restart. Should be set for an automatic restart or cleared for a one-shot operation of the timer.
2–11	—	Reserved. These bits should be written with zeros.
12–15	**TN**	Timer number. A value from 0–15 signifying which timer to use—an offset into the timer table entries.
16–31	**TP**	Timer period. The 16-bit timeout value of the timer is zero-based. The minimum value is 1 and is programmed by writing 0x0000 to the timer period.The maximum value of the timer is 65,536 and is programmed by writing 0xFFFF.

13.6.3 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.

13.6.4 RISC Timer Event Register (RTER)/Mask Register (RTMR)

The RTER is used to report events recognized by the 16 timers and to generate interrupts. RTER can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values.

The RISC timer mask register (RTMR) is used to enable interrupts that can be generated in the RTER. Setting an RTMR bit enables the corresponding interrupt in the RTER; clearing a bit masks the corresponding interrupt. An interrupt is generated only if the RISC timer table bit is set in the SIU interrupt mask register; see Section 4.3.1.5, “SIU Interrupt Mask Registers (SIMR_H and SIMR_L).”

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR 15	TMR 14	TMR 13	TMR 12	TMR 11	TMR 10	TMR 9	TMR 8	TMR 7	TMR 6	TMR 5	TMR 4	TMR 3	TMR 2	TMR 1	TMR 0
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x119D6 (RTER)/0x119DA (RTMR)															

Figure 13-11. RISC Timer Event Register (RTER)/Mask Register (RTMR)

13.6.5 SET TIMER Command

The SET TIMER command is used to enable, disable, and configure the 16 timers in the RISC timer table and is issued to the CPCR. This means the value 0x29E1008 should be written to CPCR. However, before writing this value, the user should program the TM_CMD fields. See Section 13.6.2, “RISC Timer Command Register (TM_CMD).”

13.6.6 RISC Timer Initialization Sequence

The following sequence initializes the RISC timers:

1. Configure RCCR to determine the preferred tick interval for the entire timer table. The TIME bit is normally set at this time but can be set later if all RISC timers need to be synchronized.
2. Determine the maximum number of timers to be located in the timer table. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with $4 \times n$ bytes available, where *n* is the number of timers. If *n* is less than 16, use timer 0 through timer *n*–1 to save space.
3. Clear the TM_CNT field in the RISC timer table parameter RAM to show how many ticks elapsed since the RISC internal timer was enabled. This step is optional.
4. Clear RTER, if it is not already cleared. Write ones to clear this register.
5. Configure RTMR to enable the timers that should generate interrupts. Ones enable interrupts.
6. Set the RISC timer table bit in the SIU interrupt mask register (SIMR_L[RTT]) to generate interrupts to the system. The SIU interrupt controller may require other initialization not mentioned here.
7. Configure the TM_CMD field of the RISC timer table parameter RAM. At this point, determine whether a timer is to be enabled or disabled, one-shot or restart, and what its timeout period should be. If the timer is being disabled, the parameters (other than the timer number) are ignored.
8. Issue the SET TIMER command by writing 0x29E1_0008 to the CPCR.
9. Repeat the preceding two steps for each timer to be enabled or disabled.

13.6.7 RISC Timer Initialization Example

The following sequence initializes RISC timer 0 to generate an interrupt approximately every second using a 133-MHz general system clock:

1. Write 111111 to RCCR[TIMEP] to generate the slowest clock. This value generates a tick every 65,536 clocks, which is every 485 µs at 133 MHz.
2. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with 4 bytes available. Assuming the beginning of dual-port RAM is available, write 0x0000 to TM_BASE.
3. (Optional) Write 0x0000 to the TM_CNT field in the RISC timer table parameter RAM to see how many ticks elapsed since the RISC internal timer was enabled.
4. Write 0xFFFF to the RTER to clear any previous events.
5. Write 0x0001 to the RTMR to enable RISC timer 0 to generate an interrupt.
6. Write 0x0002_0000 to the SIU interrupt mask register (SIMR_L) to allow the RISC timers to generate a system interrupt. Initialize the SIU interrupt configuration register.
7. Write 0xC000_080D to the TM_CMD field of the RISC timer table parameter RAM. This enables RISC timer 0 to timeout after 2,061(decimal) ticks of the timer. The timer automatically restarts after it times out.
8. Write 0x29E1_0008 to the CPCR to issue the SET TIMER command.
9. Set RCCR[TIME] to enable the RISC timer to begin operation.

13.6.8 RISC Timer Interrupt Handling

The following sequence describes what normally would occur within an interrupt handler for the RISC timer tables:

1. Once an interrupt occurs, read RTER to see which timers have caused interrupts. The RISC timer event bits are usually cleared by this time.
2. Issue additional SET TIMER commands at this time or later, as preferred. Nothing needs to be done if the timer is being automatically restarted for a repetitive interrupt.
3. Clear the RTT bit in the SIU interrupt pending register (SIPNR_L).
4. Execute the RTE instruction.

13.6.9 RISC Timer Table Scan Algorithm

The CP scans the timer table once every tick. It handles each of the 16 timers at its turn and checks for other requests with higher priority to service, before handling the next one. For each valid timer in the table, the CP decrements the count and checks for a timeout. If none occurs, the CP moves to the next timer. If a timeout occurs, the CP sets the corresponding event bit in RTER. Then the CP checks to see if the timer is to be restarted and if it is, the CP leaves the timer’s valid bit set in the R_TMV location and resets the current count to the initial count. Otherwise, it clears R_TMV. Once the timer table scanning has completed, the CP updates the TM_CNT value in the RISC timer table parameter RAM and stops working on the timer tables until the next tick.

If a SET TIMER command is issued, the CP makes the appropriate modifications to the timer table and parameter RAM, but does not scan the timer table until the next tick of the internal timer. It is important to use the SET TIMER command to properly synchronize timer table modifications to the execution of the CP.

13.6.10 Using the RISC Timers to Track CP Loading

The RISC timers can be used to track CP loading. The following sequence provides a way to use the 16 RISC timers to determine if the CP ever exceeds the 96% utilization level during any tick interval. Removing the timers adds a 4% margin to the CP utilization level, but the aggressive user can use this technique to push CP performance to its limit. The user should use the standard initialization sequence and incorporate the following differences:

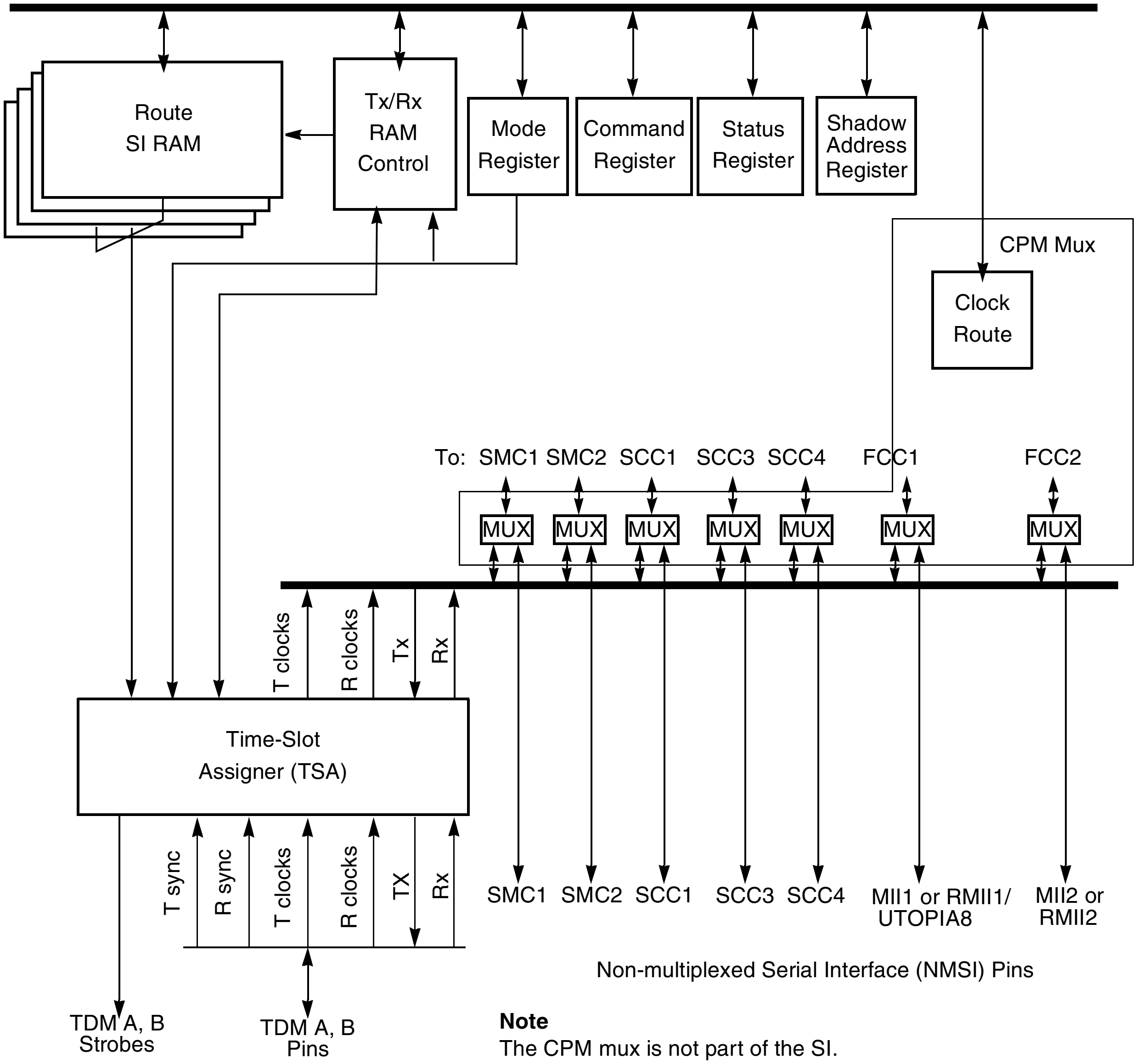
1. Program the tick of the RISC timers to be every 1,024 x 16 = 16,384 system clocks.
2. Disable RISC timer interrupts, if preferred.
3. Using the SET TIMER command, initialize all 16 RISC timers to have a timer period of 0xFFFF, which equates to 65,536.
4. Program one of the four general-purpose timers to increment once every tick. The general-purpose timer should be free-running and should have a timeout of 65,536.
5. After a few hours of operation, compare the general-purpose timer to the current count of RISC timer 15. If it is more than two ticks different from the general-purpose timer, the CP has, during some tick interval, exceeded the 96% utilization level.

NOTE

General-purpose timers are up counters, but RISC timers are down counters. The user should take this under consideration when comparing timer counts.

Chapter 14
Serial Interface with Time-Slot Assigner

Figure 14-1 shows the block diagram of the time-slot assigner (TSA). One SI block is present in the MPC8272 (SI2). It can be programmed to handle two TDM lines concurrently. TDM channels on SI2 are referred to as TDMa2 and TDMb2.



Note
The CPM mux is not part of the SI.

Figure 14-1. SI Block Diagram

If the TSA is not used as intended, it can be used to generate complex wave forms on dedicated output pins. For instance, it can program these pins to implement stepper motor control or variable-duty cycle and period control on-the-fly.

14.1 Features

SI has the following features:

- Can connect to two independent TDM channels. Each TDM can be one of the following:
  - T1 or E1 line
  - Integrated services digital network primary rate (PRI)
  - An ISDN basic rate–interchip digital link (IDL) channel in up to four TDM channels—each IDL channel requires support from a separate SCC
  - An ISDN basic rate–general circuit interface (GCI) in up to two TDMs channels—each GCI channel requires support from a separate SMC
  - E3 or DS3 clear channel
  - User-defined interfaces
- Independent, programmable transmit and receive routing paths
- Independent transmit and receive frame syncs allowed
- Independent transmit and receive clocks allowed
- Selection of rising/falling clock edges for the frame sync and data bits
- Supports 1× and 2× input clocks (1 or 2 clocks per data bit)
- Selectable delay (0–3 bits) between frame sync and frame start
- Four programmable strobe outputs and two clock output pins
- 1- or 8-bit resolution in routing, masking, and strobe selection
- Supports frames up to 16,384 bits long
- Internal routing and strobe selection can be dynamically programmed
- Supports automatic echo and loopback mode for each TDM
- Maximum TDM frequency is serial-dependent:

  - For FCCs transparent
  - For FCCs HDLC

$$\frac{\text{CPM Clock}}{4}$$

  - For all other serials

$$\frac{\text{CPM Clock}}{3}$$

14.2 Overview

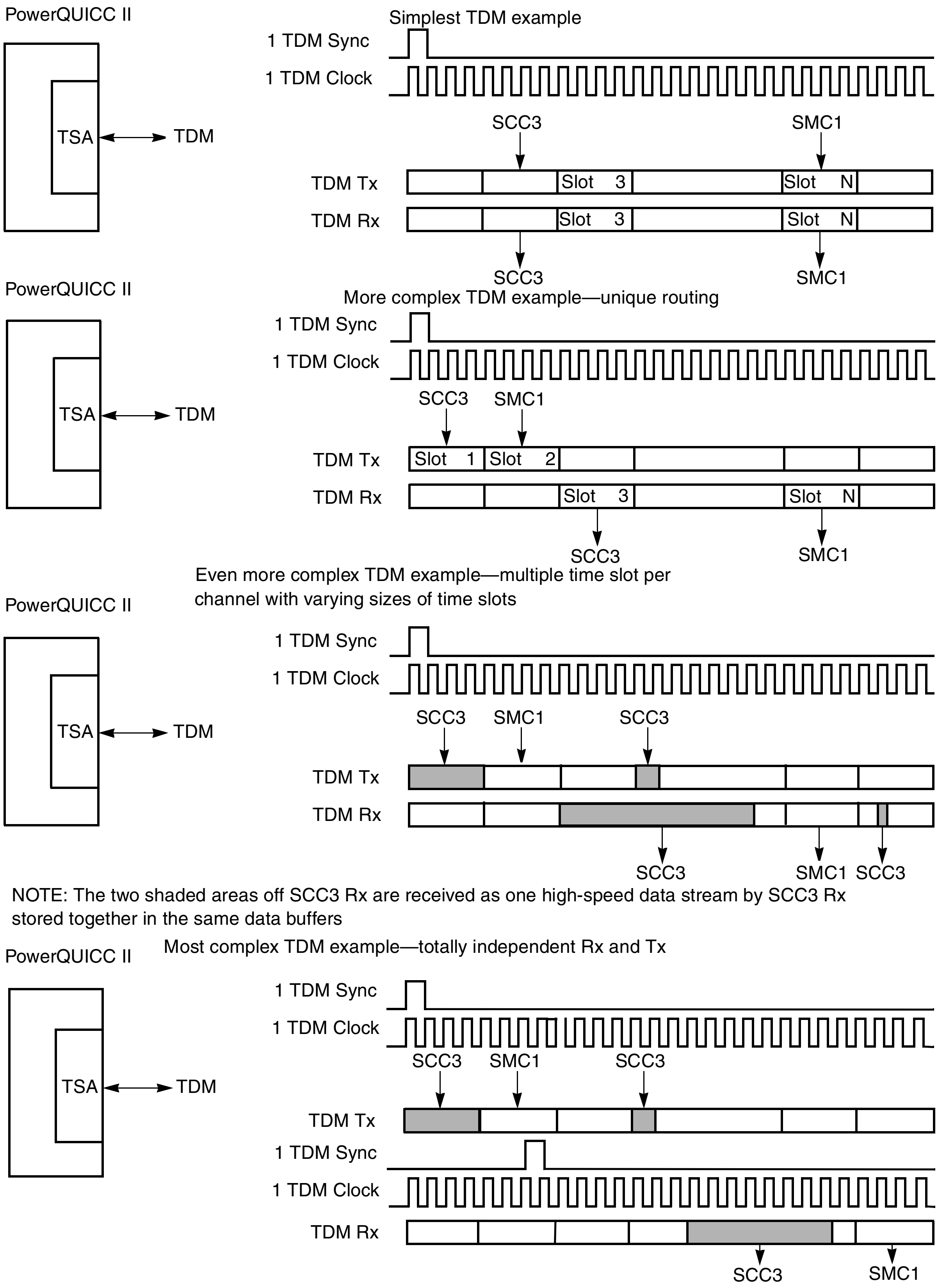
The TSA implements both internal route selection and time-division multiplexing (TDM) for multiplexed serial channels. The TSA supports the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates. The two popular ISDN basic rate buses (interchip digital link (IDL) and general-circuit interface (GCI), also known as IOM-2, are supported.

Because the SI supports two TDMs, it is possible to simultaneously support a combination of two T1 or E1 lines, and basic rate or primary rate ISDN channels.

The TDM channel can support E3 or DS-3 rates as a clear channel in a serial interface (clock ratio 1/4).

TSA programming is independent of the protocol used. The serial controllers can be programmed for any synchronous protocol without affecting TSA programming. The TSA simply routes programmed portions of the received data frame from the TDM pins to the target controller, while the target controller handles the received data in the actual protocol.

In its simplest mode, the TSA identifies the frame using one sync pulse and one clock signal provided externally by the user. This can be enhanced to allow independent routing of the receive and transmit data on the TDM. Additionally, the definition of a time-slot need not be limited to 8 bits or even to a single contiguous position within the frame. Finally, the user can provide separate receive and transmit syncs as well as clocks. Figure 14-2 shows example TSA configurations ranging from the simplest to the most complex.



Simplest TDM example

More complex TDM example—unique routing

Even more complex TDM example—multiple time slot per channel with varying sizes of time slots

NOTE: The two shaded areas off SCC3 Rx are received as one high-speed data stream by SCC3 Rx stored together in the same data buffers

Most complex TDM example—totally independent Rx and Tx

Figure 14-2. Various Configurations of a Single TDM Channel

At its most flexible, the TSA can provide two separate TDM channels, each with independent receive and transmit routing assignments and independent sync pulse and clock inputs. Thus, the TSA can support four, independent, half-duplex TDM sources, two in reception and two in transmission, using four sync inputs and four clock inputs. Figure 14-3 shows a dual-channel example.

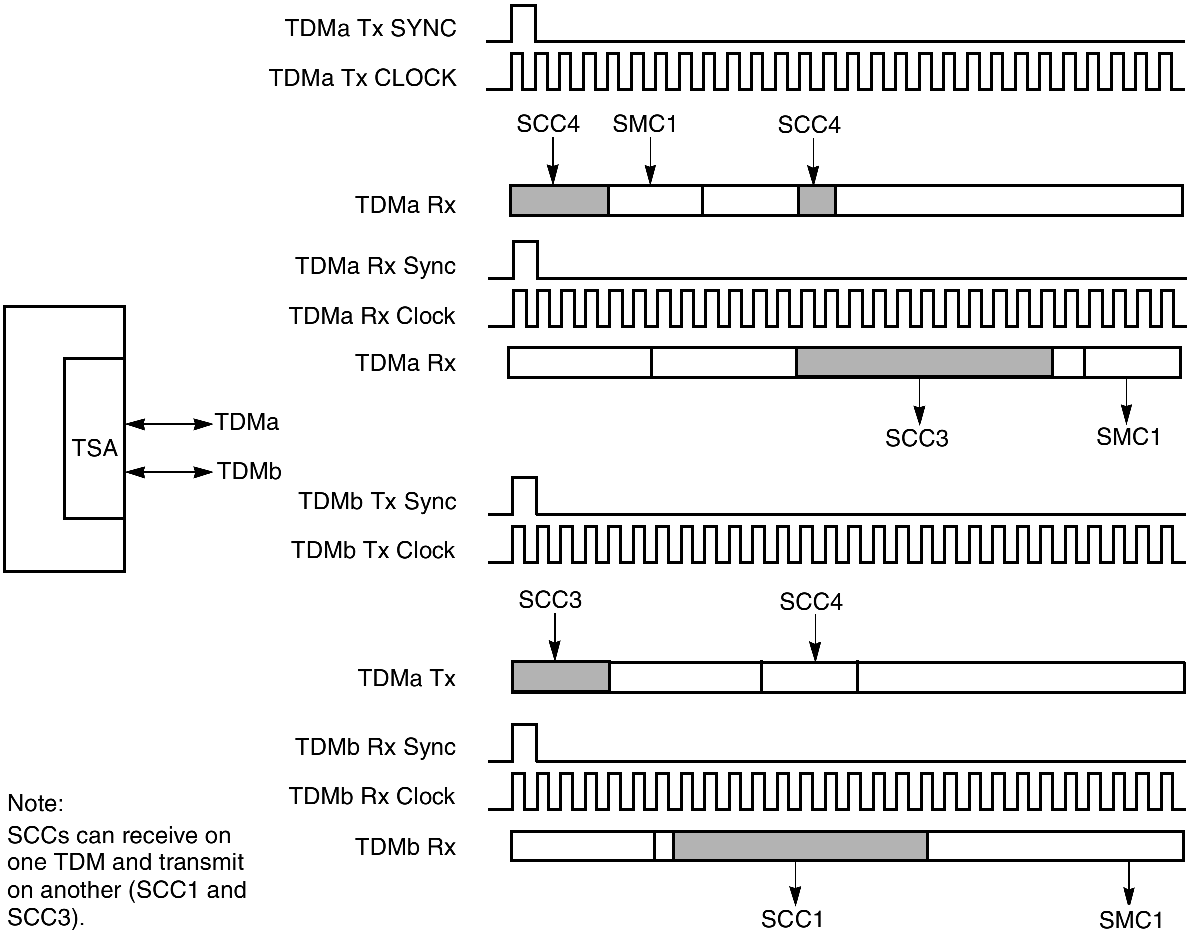


Figure 14-3. Dual TDM Channel Example

In addition to channel programming, the TSA supports up to four strobe outputs that may be asserted on a bit or byte basis. These strobes are completely independent from the channel routing used by the SCCs and SMCs. The strobe outputs are useful for interfacing to other devices that do not support the multiplexed interface or for enabling/disabling three-state I/O buffers in a multiple-transmitter architecture. Notice that open-drain programming on the TXDx pins that supports a multiple-transmitter architecture occurs in the parallel I/O block. These strobes can also be used for generating output wave forms to support such applications as stepper-motor control.

Most TSA programming is done in the two 256- × 16-bit SI2 RAM. The SI2 RAM is directly accessible by the core in the internal register section of the PowerQUICC II and is not associated with the dual-port RAM. The SI2 RAM is always used to program the transmit routing; the other is always used to program the receive routing. SI2 RAM can be used to define the number of bits/bytes to be routed to the FCC, SCC, or SMC and determine when external strobes are to be asserted and negated.

The size of the SI2 RAM available for time-slot programming depends on the user's configuration. The user defines how many of the 256 entries are related to each TDM. The resolution of the division is by fractions of 32. If on-the-fly changes are allowed, the SI2 RAM entries are reduced according to the user's programming. The maximum frame length that can be supported in any configuration is 16,384 bits.

The maximum external serial clock that may be an input to the TSA is CPM CLK/3.

The SI supports two testing modes: echo and loopback.

- Echo mode provides a return signal from the physical interface by retransmitting the signal it has received. The physical interface echo mode differs from the individual FCC or SCC echo mode in that it can operate on the entire TDM signal rather than just on a particular serial channel.
- Loopback mode causes the physical interface to receive the same signal it is sending. The SI loopback mode checks more than the individual serial loopback; it checks both the SI and the internal channel routes.

NOTE

The flexibility described in the preceding section can be applied to each of the two TDM channels and to all serial interfaces independently.

14.3 Enabling Connections to TSA

Each serial interface can be independently enabled to connect to one of the following: TSA, UTOPIA, MII, or dedicated external pins. Note the following:

- Each FCC can be connected to a dedicated MII or one of two TDMs. FCC1 can also be connected to a 8-bit UTOPIA level-2 interface.
- Each SCC or SMC can be connected to one of two TDMs or to its own set of pins.

The two TDMs are connected to two independent TDM interfaces. Figure 14-4 illustrates the connection between the TSA and the serial interfaces. The connection is made by programming the CPM mux. See Chapter 15, "CPM Multiplexing." Once the connections are made, the exact routing decisions are made in the SI2 RAM.

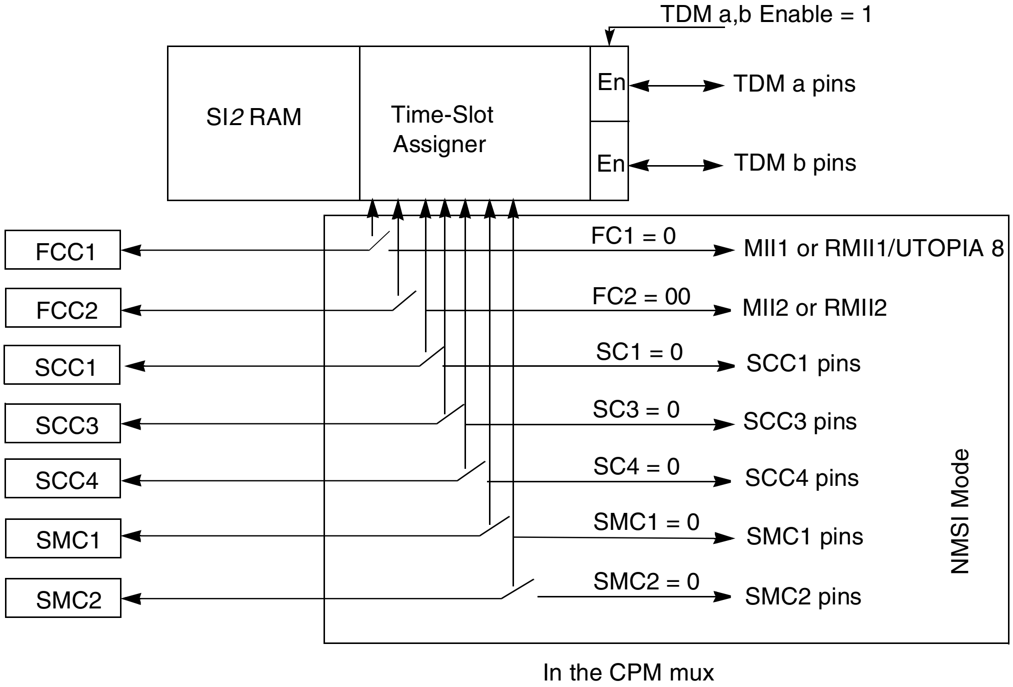


Figure 14-4. Enabling Connections to the TSA

14.4 Serial Interface RAM

The SI has a transmit RAM and a receive RAM, each with four banks of 64 half-word entries that enable it to control TDM channel routing to all serial devices. The SI2 RAM is uninitialized after power-on reset; unwanted results can occur if the user does not program it before enabling the multiplexed channels.

Each 16-bit SI RAM entry defines the routing of 1–8 bits or bytes at a time. In addition to the routing, up to four strobe pins (logic OR of four strobes in the transmit RAM and four in receive RAM) can be asserted according to the programming of the RAMs. The four SI2 RAM banks can be configured in many different ways to support various TDM channels. The user can define the size of each SI2 RAM that is related to a certain TDM channel by programming the starting bank of that TDM. Programming the starting shadow bank address, described in Section 14.5.3, “SI2 RAM Shadow Address Register (SI2RSR),” determines whether this RAM has a shadow for changing SI2 RAM entries while the TDM channel is active. This reduces the number of available SI2 RAM entries for that TDM.

14.4.1 Single Multiplexed Channel with Static Frames

The example in Figure 14-5 shows one of many possible settings. With this configuration, the SI2 RAM has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. This configuration should be chosen only when one TDM is required and the routing on that TDM does not need to be dynamically changed. The number of entries available in the SI2 RAM is determined by the user.

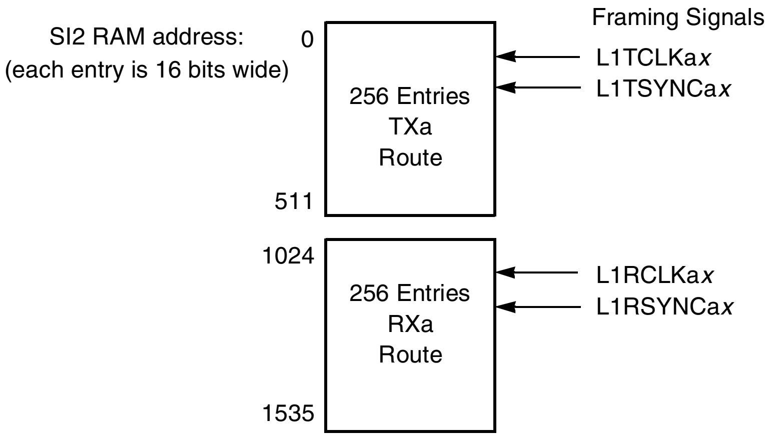


Figure 14-5. Single TDM Channel with Static Frames and Independent Rx and Tx Routes

14.4.2 Single Multiplexed Channel with Dynamic Frames

In the configuration shown in Figure 14-6, a multiplexed channel has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. Each RAM has two sections: the current-route RAM and a shadow RAM for changing serial routing dynamically.

After programming the shadow RAM, the user sets SI2CMDR[CSR*xn*] for the associated channel. When the next frame sync arrives, the SI automatically exchanges the current-route RAM for the shadow RAM. See Section 14.4.5, “Static and Dynamic Routing.”

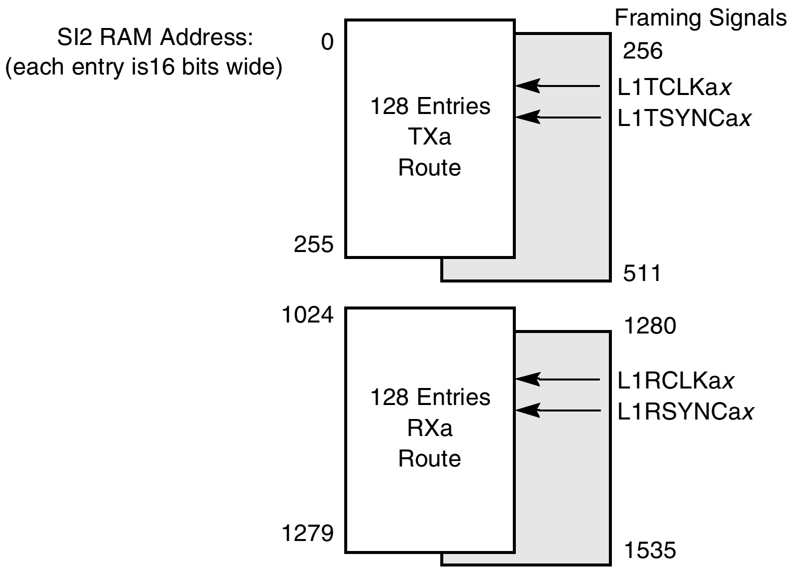


Figure 14-6. Single TDM Channel with Shadow RAM for Dynamic Route Change

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 SI2 RAM entries are reduced for every TDM channel programmed for shadow mode.

14.4.3 Programming SI2 RAM Entries

The programming of each entry in the SI2 RAM determines the routing of the serial bits (or bit groups) and the assertion of strobe outputs. Figure 14-7 shows the entry fields.

	0	1	2	3	4	5	6	7 10	11 13	14	15
Field	—	SWTR	SSEL1	SSEL2	SSEL3	SSEL4	—	CSEL	CNT	BYT	LST
R/W	R/W										
Addr	See Chapter 3, “Memory Map.”										

Figure 14-7. SI2 RAM Entry Fields

The SI2 RAM entry fields function as described in Table 14-1.

Table 14-1. SI2 RAM Entry

Bits	Name	Description
0	—	Reserved, should be cleared.
1	SWTR	Switch Tx and Rx. Valid only in the receive route RAM and ignored in the transmit route RAM. SWTR affects the operation of both L1RXD and L1TXD. SWTR is set only in special situations where the user prefers to receive data from a transmit pin and transmit data on a receive pin. For instance, where devices A and B are connected to the same TDM, each with different time-slots. Normally, there is no opportunity for stations A and B to communicate with each other directly over the TDM, because they both receive the same TDM receive data and transmit on the same TDM transmit signal. 0 Normal operation of L1TXD and L1RXD. 1 Data for this entry is sent on L1RXD and received from L1TXD. See Figure 14-8 for details.
2–5	SSELx	Strobe select. There are four strobes available that can be assigned to the receive RAM and asserted/negated with the received clock of this TDM channel (L1RCLKx). They can also be assigned to the transmit RAM and asserted/negated with the transmit clock of this TDM channel (L1TCLKx). Each bit corresponds to the value the strobe should have during this bit/byte group. There are four strobe pins for all eight strobe bits in the SI2 RAM entries, so the value on a strobe pin is the logical OR of the Rx and Tx RAM entry strobe bits. Multiple strobes can be asserted simultaneously. A strobe configured to be asserted in consecutive SI*2* RAM entries remains continuously asserted for both entries. A strobe asserted on the last entry in a table is negated after the last entry is processed. **Note:** Each strobe is changed with the corresponding RAM clock and is output only if the corresponding parallel I/O is configured as a dedicated pin. If a strobe is programmed to be asserted in more than one set of entries (the SI route entries for more then one TDM channel select the same strobe), the assertion of the strobe corresponds to the logical OR of all possible sources. This use of strobes is not useful for most applications. A given strobe should be selected in only one set of SI2 RAM entries.
6	—	Reserved, should be cleared.

Table 14-1. SI2 RAM Entry (continued)

Bits	Name	Description
7–10	CSEL	Channel select. 0000 The bit/byte group is not supported by the PowerQUICC II. The transmit data pin is three-stated and the receive data pin is ignored. 0001 The bit/byte group is routed to SCC1. 0010 Reserved 0011 The bit/byte group is routed to SCC3. 0100 The bit/byte group is routed to SCC4. 0101 The bit/byte group is routed to SMC1. 0110 The bit/byte group is routed to SMC2. 0111 The bit/byte group is not supported by the PowerQUICC II. This code is also used in SCIT mode as the D channel grant. See Section 14.7.2.2, “SCIT Programming.” 1000 Reserved 1001 The bit/byte group is routed to FCC1. 1010 The bit/byte group is routed to FCC2. 1011 Reserved 11xx Reserved
11–13	CNT	Count. Indicates the number of bits/bytes (according to the BYT bit) that the routing and strobe select of this entry controls. 000 = 1 bit/byte; 111= 8 bits/bytes.
14	BYT	Byte resolution 0 Bit resolution. The CNT value indicates the number of bits in this group. 1 Byte resolution. The CNT value indicates the number of bytes in this group.
15	LST	Last entry in the RAM. Whenever SI2 RAM is used, LST must be set in one of the Tx or Rx entries of each group. Even if all entries of a group are used, this bit must still be set in the last entry. 0 Not the last entry in this section of the route RAM. 1 Last entry in this RAM. After this entry, the SI waits for the sync signal to start the next frame. **Note:** There must be only an even number of entries in an SI2 RAM frame, because LST is active only in odd-numbered entries (assuming the entry count starts with 0). Therefore, to obtain an even number of entries, an entry may need to be split into two entries. Also note that, to avoid errors in switching to and from shadow SI RAM, the last entry in SI RAM should not be programmed to 1-bit resolution (i.e. CNT = 000 and BYT = 0).

Figure 14-8 shows how SWTR can be used.

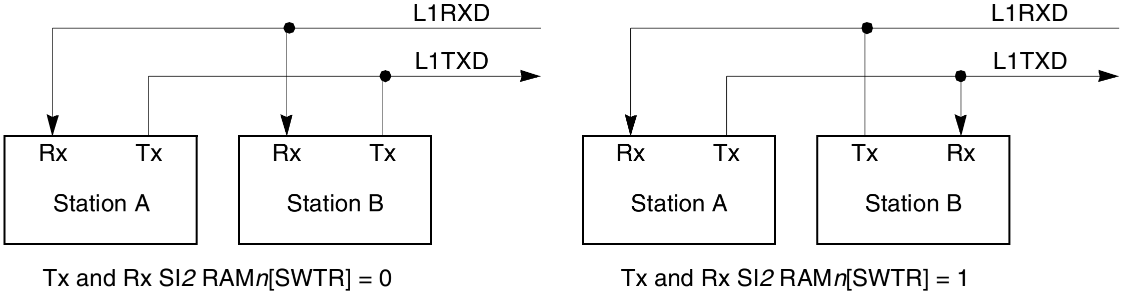


Figure 14-8. Using the SWTR Feature

The SWTR option lets station B listen to transmissions from station A and send data to station A. To do this, station B would set SWTR in its receive route RAM. For this entry, receive data is taken from the L1TXD pin and data is sent on the L1RXD pin. If the user wants to listen only to station A transmissions and not send data on L1RXD, the CSEL bits in the corresponding transmit route RAM entry should be cleared to prevent transmission on the L1RXD pin.

Station B can transmit data to station A by setting the SWTR bit of the entry in its receive route RAM. Data is sent on L1RXD rather than L1TXD, according to the transmit route RAM. Note that this configuration could cause collisions with other data on L1RXD unless an available (quiet) time slot is used. To transmit on L1RXD and not receive data on L1TXD, clear the CSEL bits in the receive route RAM.

NOTE

> If the transmit and receive sections of the TDM do not use a common clock source, the SWTR feature can cause erratic behavior.

14.4.4 SI2 RAM Programming Example

This example shows how to program the RAM to support the 10-bit IDL bus. Figure 14-23 shows the 10-bit IDL bus format. In this example, the TSA supports the B1 channel with SCC3, the D channel with SCC1, the first 4 bits of the B2 channel with an external device (using a strobe to enable the external device), and the last 4 bits of B2 with SMC1. Additionally, the TSA marks the D channel with another strobe signal.

First, divide the frame from the start (the sync) to the end of the frame according to the support that is required:

- 8 bits (B1)—SCC3
- 1 bit (D)—SCC1 + strobe 1
- 1 bit—no support
- 4 bits (B2)—strobe 2
- 4 bits (B2)—SMC1
- 1 bit (D)—SCC1 + strobe 1

Each of these six divisions can be supported by a single SI2 RAM entry. Thus, six SI2 RAM entries are needed. See Table 14-2.

Table 14-2. SI2 RAM Entry Descriptions

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8-bit SCC2
1	0	1000	0001	000	0	0	1-bit SCC1 strobe1
2	0	0000	0000	000	0	0	1-bit no support
3	0	0100	0000	011	0	0	4-bit strobe2
4	0	0000	0101	011	0	0	4-bit SMC1
5	0	1000	0001	000	0	1	1-bit SCC1 strobe1

NOTE

IDL requires the same routing for both receive and transmit., Therefore, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI2 RAM. Then SI2MR[CRTx] can be used to instruct the SI2 RAM to use the same clock and sync to simultaneously control both sets of SI2 RAM entries.

14.4.5 Static and Dynamic Routing

SI2 RAM has two operating modes for the TDMs:

- Static routing—The number of SI2 RAM entries is determined by the banks the user relates to the corresponding TDM and is divided into two parts (Rx and Tx). The following sequence must be followed to program the routing entries.
  - All serial devices connected to the TSA must be disabled.
  - SI routing can be modified.
  - All appropriate serial devices connected to the TSA must be reenabled.
- Dynamic routing—A TDM’s routing definition can be modified while FCCs, SCCs, or SMCs are connected to the TDM. The number of SI2 RAM entries is determined by the banks the user relates to the corresponding TDM channel and is divided into four parts (Rx, Rx shadow, Tx, and Tx shadow).

Dynamic changes divide portions of the SI2 RAM into current-route and shadow RAM. Once the current-route RAM is programmed, the TSA and SI channels are enabled, and TSA operation begins. When a change in routing is required, the shadow RAM must be programmed with the new route and SI2CMDR[CSR*xn*] must be set. As a result, as soon as the corresponding sync arrives the SI exchanges the shadow RAM with the current-route RAM and resets CSR*xn* to indicate that the operation is complete. At this time, the user may change the routing again. Notice that the original current-route RAM is now the shadow RAM and vice versa. Figure 14-9 shows an example of the shadow RAM exchange process for two TDM channels both with half of the RAM as a shadow.

If for instance one TDM with dynamic changes is programmed to own all four banks, and the shadow is programmed to the last two banks, the initial current-route RAM addresses in the SI2 RAM are as follows:

- 0–255: TXa route
- 1024–1279: RXa route

The initial shadow RAMs are at addresses:

- 256–511: TXa route
- 1280–1535: RXa route

The user can read any RAM at any time, but for proper SI operation the user must not attempt to write the current-route RAM. The SI2 status register (SI2STR) can be read to find out which part of the RAM is the current-route RAM. The user can also externally connect one of the strobes to an interrupt pin to generate an interrupt on a particular SI2 RAM entry starting or ending execution by the TSA.

NOTE

The current-route and shadow SI RAMs of a given TDM*x* should be contiguous; that is, the current-route and shadow SI RAMs of differing TDM*x* should not be interleaved. An example is shown in Figure 14-9.

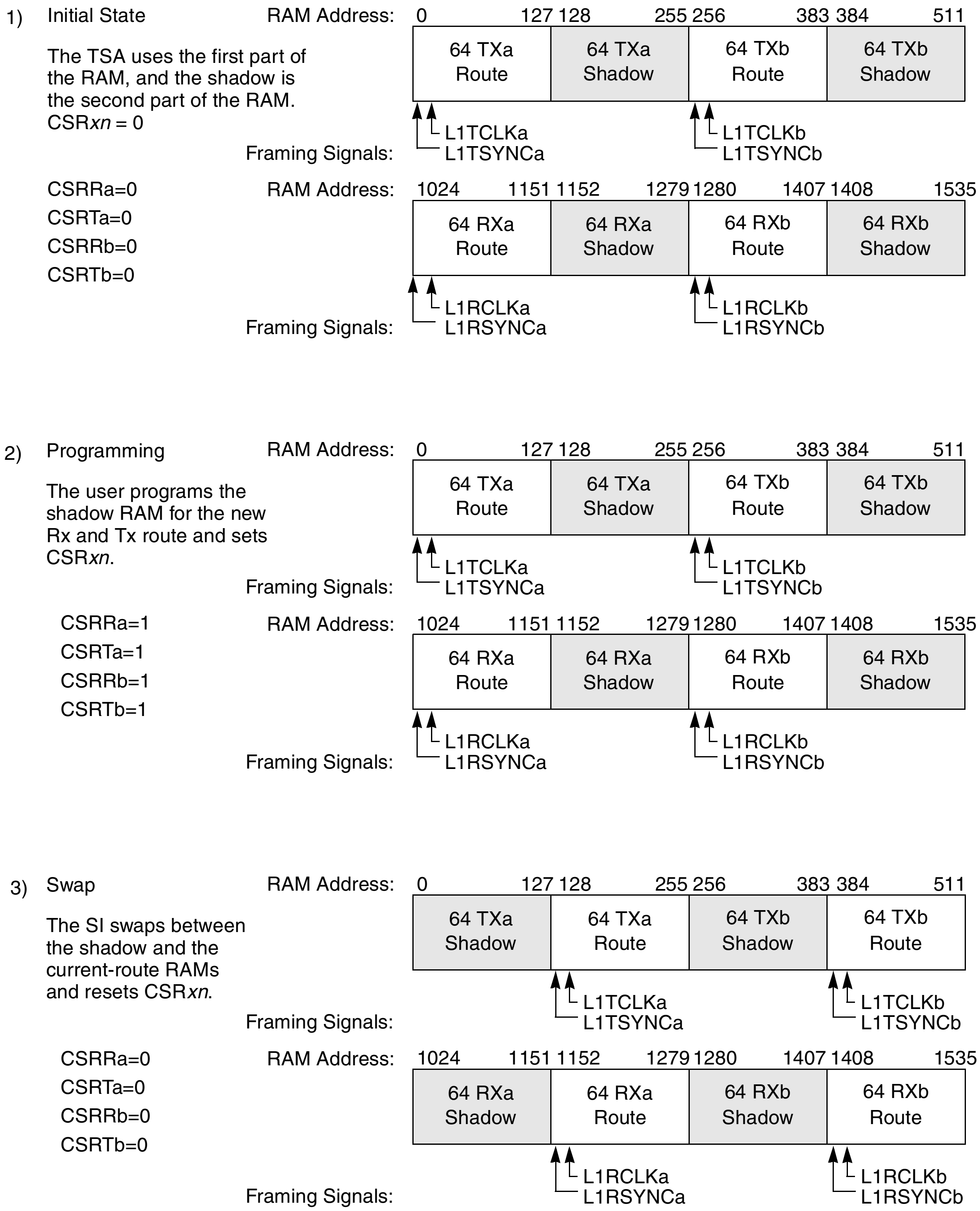


Figure 14-9. Example: SI2 RAM Dynamic Changes, TDMa and b, Same SI2 RAM Size

14.5 Serial Interface Registers

The serial interface registers are described in the following sections.

14.5.1 SI2 Global Mode Register (SI2GMR)

The SI2 global mode register (SI2GMR), shown in Figure 14-10, defines the activation state of the TDM channels for SI2.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>Field</td><td>—</td><td>—</td><td>STZB</td><td>STZA</td><td>—</td><td>—</td><td>ENB</td><td>ENA</td></tr>
<tr><td>Reset</td><td colspan="8">0000_0000</td></tr>
<tr><td>R/W</td><td colspan="8">R/W</td></tr>
<tr><td>Addr</td><td colspan="8">0x11B48 (SI2GMR)</td></tr>
</table>

Figure 14-10. SI Global Mode Register (SI2GMR)

Table 14-3 describes SI2GMR.

Table 14-3. SI2GMR Field Descriptions

Bit	Name	Description
0,1,4,5	—	Reserved, should be cleared.
2, 3	STZx	Program L1TXDx to zero for TDM a or b 0 Normal operation 1 L1TXDx = 0 until serial clocks are available, which is useful for GCI activation. See Section 14.7.1, "SI GCI Activation/Deactivation Procedure."
6, 7	ENx	Enable TDMx. Note that enabling a TDM is the last step in initialization. 0 TDM channel x is disabled. The SI2 RAMs and routing for TDMx are in a state of reset, but all other SI functions still operate. 1 All TDMx functions are enabled.

14.5.2 SI2 Mode Register (SI2MR)

Two SI mode registers (SI2MR), are shown in Figure 14-11, one for each TDM channel (SI2AMR, and SI2BMR). They are used to define SI operation modes and allow the user (with SI2 RAM) to support any or all of the ISDN channels independently when in IDL or GCI mode. Any extra serial channel can then be used for other purposes.

<table>
<tr><th></th><th>0</th><th>1</th><th></th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Field</td><td>—</td><td colspan="3">SADx</td><td colspan="2">SDMx</td><td colspan="2">RFSDx</td><td>DSCx</td><td>CRTx</td><td>SLx</td><td>CEx</td><td>FEx</td><td>GMx</td><td colspan="2">TFSDx</td></tr>
<tr><td>Reset</td><td colspan="16">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="16">R/W</td></tr>
<tr><td>Addr</td><td colspan="16">0x11B40 (SI2AMR), 0x11B42 (SI2BMR)</td></tr>
</table>

Figure 14-11. SI Mode Register (SI2MR)

Table 14-4 describes SI2MR fields.

Table 14-4. SI2MR Field Descriptions

Bits	Name	Description
0	—	Reserved. Should be cleared.
1–3	SADx	Starting bank address for the RAM of TDM a or b These three bits define the starting bank address of the SI2 RAM section that belongs to TDMx channel. **Note:** As noted previously, the SI2 RAM contains four banks of 64 entries for receive and four banks of 64 entries for transmit. The starting bank address of each TDM can be programmed with a granularity of 32 entries. The user can put the shadow RAM section of the same TDM on the same bank, but the user cannot put two different TDMs on the same bank. The last entry of a certain TDM is determined by the LST bit in the SI2 RAM entry. The user must set LST within the entries of SI2 RAM blocks for every TDM used, that is, before the starting address of the next TDM. 000 First bank, first 32 entries 001 First bank, second 32 entries 010 Second bank, first 32entries 011 Second bank, second 32 entries 100 Third bank, first 32 entries 101 Third bank, second 32 entries 110 Fourth bank, first 32 entries 111 Fourth bank, second 32 entries
4–5	SDMx	SI Diagnostic mode for TDM a or b 00 Normal operation. 01 Automatic echo. In this mode, the TDM transmitter automatically retransmits the TDM received data on a bit-by-bit basis. The receive section operates normally, but the transmit section can only retransmit received data. In this mode, the L1GRx line is ignored. 10 Internal loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The receiver and transmitter operate normally. The data appears on the L1TXDx pin and in this mode, $\overline{\text{L1RQ}x}$ is asserted normally. The L1GRx line is ignored. 11 Loopback control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The transmitter output (L1TXDx) and $\overline{\text{L1RQ}x}$ are inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines. **Note:** In modes 01, 10, and 11, the receive and transmit clocks should be identical.
6–7	RFSDx	Receive frame sync delay for TDM a or b. Determines the number of clock delays between the receive sync and the first bit of the receive frame. Even if CRTx is set, these bits do not control the delay for the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync; use for GCI. 01 1-bit delay. Use for IDL 10 2-bit delay 11 3-bit delay Figure 14-12 and Figure 14-13 show how these bits are used. **Note:** If frame sync delay is not used and if a frame sync is issued early during the last bit of the previous frame, data corruption can occur on all subsequent frames. To avoid this problem, program a 1-, 2-, or 3-bit sync delay. When using bit delay of 0, ensure that during the last phase (negative edge) of the last serial clock of the last SIRAM entry, the sync signal is deasserted. No sync signal should be asserted early, for example, during the last clock of the frame.

Table 14-4. SI2MR Field Descriptions (continued)

Bits	Name	Description
8	DSCx	Double speed clock for TDM a or b. Some TDMs, such as GCI, define the input clock to be twice as fast as the data rate and this bit controls this option. 0 The channel clock (L1RCLKx and/or L1TCLKx) is equal to the data clock. Use for IDL and most TDM formats. 1 The channel clock rate is twice the data rate. Use for GCI. **Note:** When an SI is in 2X mode (DSC=1), the SI does not ignore sync signals asserted in the last phase of the last clock cycle of the frame.
9	CRTx	Common receive and transmit pins for TDM a or b. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx can be used for their alternate functions. 0 Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing and the transmit section uses L1TCLKx and L1TSYNCx for framing. 1 Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. Use for IDL and GCI. RFSD and TFSD are independent of one another in this mode.
10	SLx	Sync level for TDM a or b. 0 The L1RSYNCx and L1TSYNCx signals are active on logic “1”. 1 The L1RSYNCx and L1TSYNCx signals are active on logic “0”.
11	CEx	Clock edge for TDM a or b. The function depends on DSCx. When DSCx = 0: 0 The data is sent on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is sent on the falling edge of the clock and received on the rising edge. When DSCx = 1: 0 The data is sent on the rising edge of the clock and received on the rising edge. 1 The data is sent on the falling edge of the clock and received on the falling edge (use for GCI). See Figure 14-14 and Figure 14-15.
12	FEx	Frame sync edge for TDM a or b. Determines whether L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock. See Figure 14-13, Figure 14-14, Figure 14-15, and Figure 14-16. 0 Falling edge. Use for IDL and GCI. 1 Rising edge.
13	GMx	Grant mode for TDM a or b. 0 GCI/SCIT mode. The GCI/SCIT D channel grant mechanism for transmission is internally supported. The grant is one bit from the receive channel. This bit is marked by programming the channel select bits of the SI2 RAM with 0111 to assert an internal strobe on it. See Section 14.7.2.2, “SCIT Programming.” 1 IDL mode. A grant mechanism is supported if the corresponding CMXSCR[GR*x*] bit is set. The grant is a sample of L1GRx while L1RSYNCx is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel. See Section 14.6.2, “IDL Interface Programming.”
14–15	TFSDx	Transmit frame sync delay for TDM a or b. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. See Figure 14-16 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay 10 2-bit delay 11 3-bit delay **Note:** If frame sync delay is not used and if a frame sync is issued early during the last bit of the previous frame, data corruption can occur on all subsequent frames. To avoid this problem, program a 1-, 2-, or 3-bit sync delay.

Figure 14-12 shows the one-clock delay from sync to data when *x*FSD = 01.

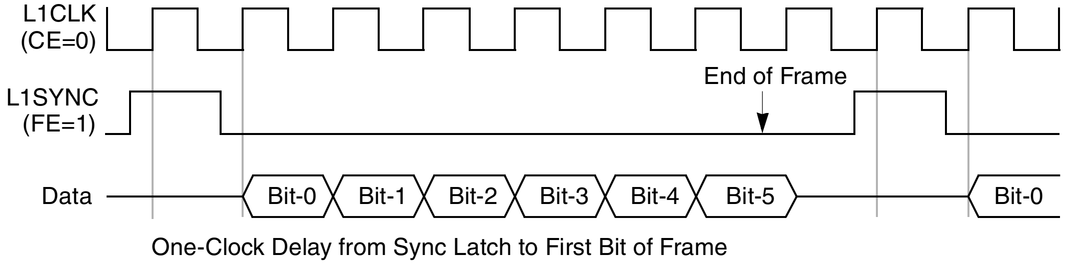


Figure 14-12. One-Clock Delay from Sync to Data (*x*FSD = 01)

Figure 14-13 shows the elimination of the single-clock delay shown in Figure 14-12 by clearing *x*FSD.

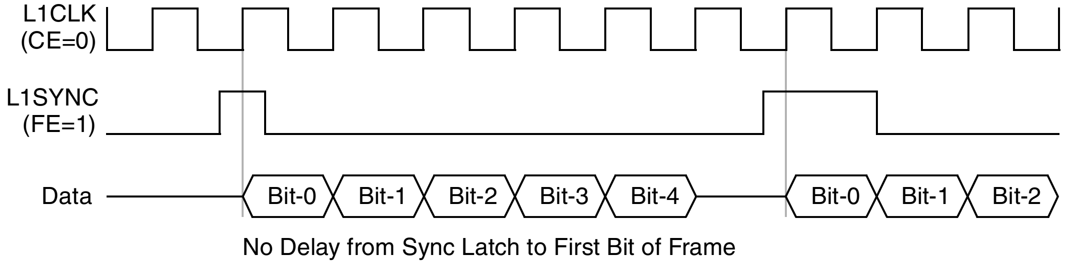


Figure 14-13. No Delay from Sync to Data (*x*FSD = 00)

Figure 14-14 shows the effects of changing FE when CE = 1 with a 1-bit frame sync delay.

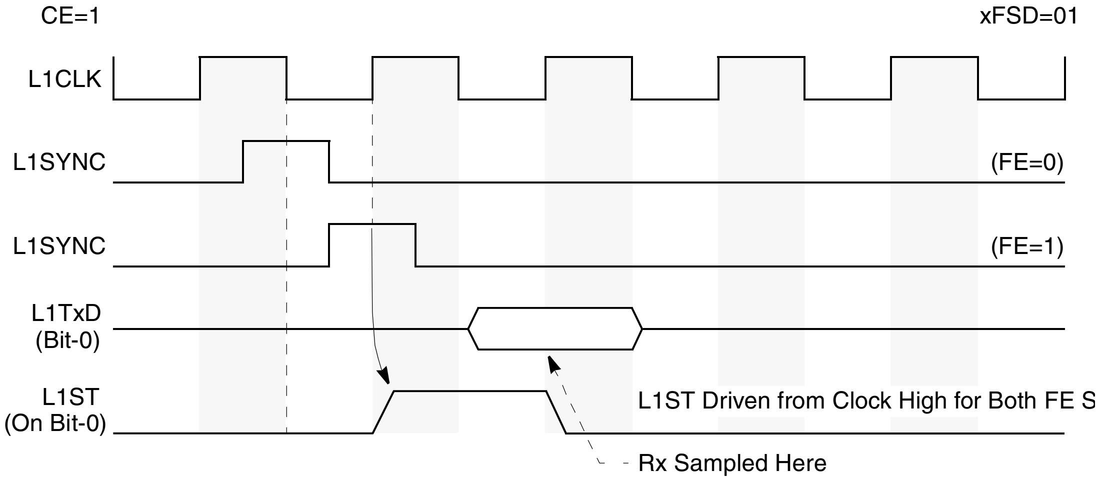


Figure 14-14. Falling Edge (FE) Effect when CE = 1 and *x*FSD = 01

Figure 14-15 shows the effects of changing FE when CE = 0 with a 1-bit frame sync delay.

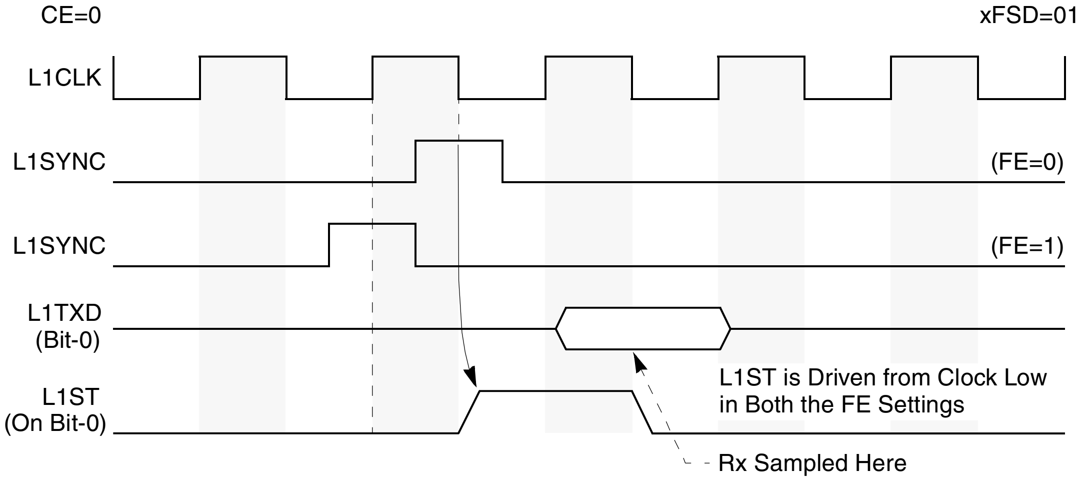


Figure 14-15. Falling Edge (FE) Effect when CE = 0 and *x*FSD = 01

Figure 14-16 shows the effects of changing FE when CE = 1 with no frame sync delay.

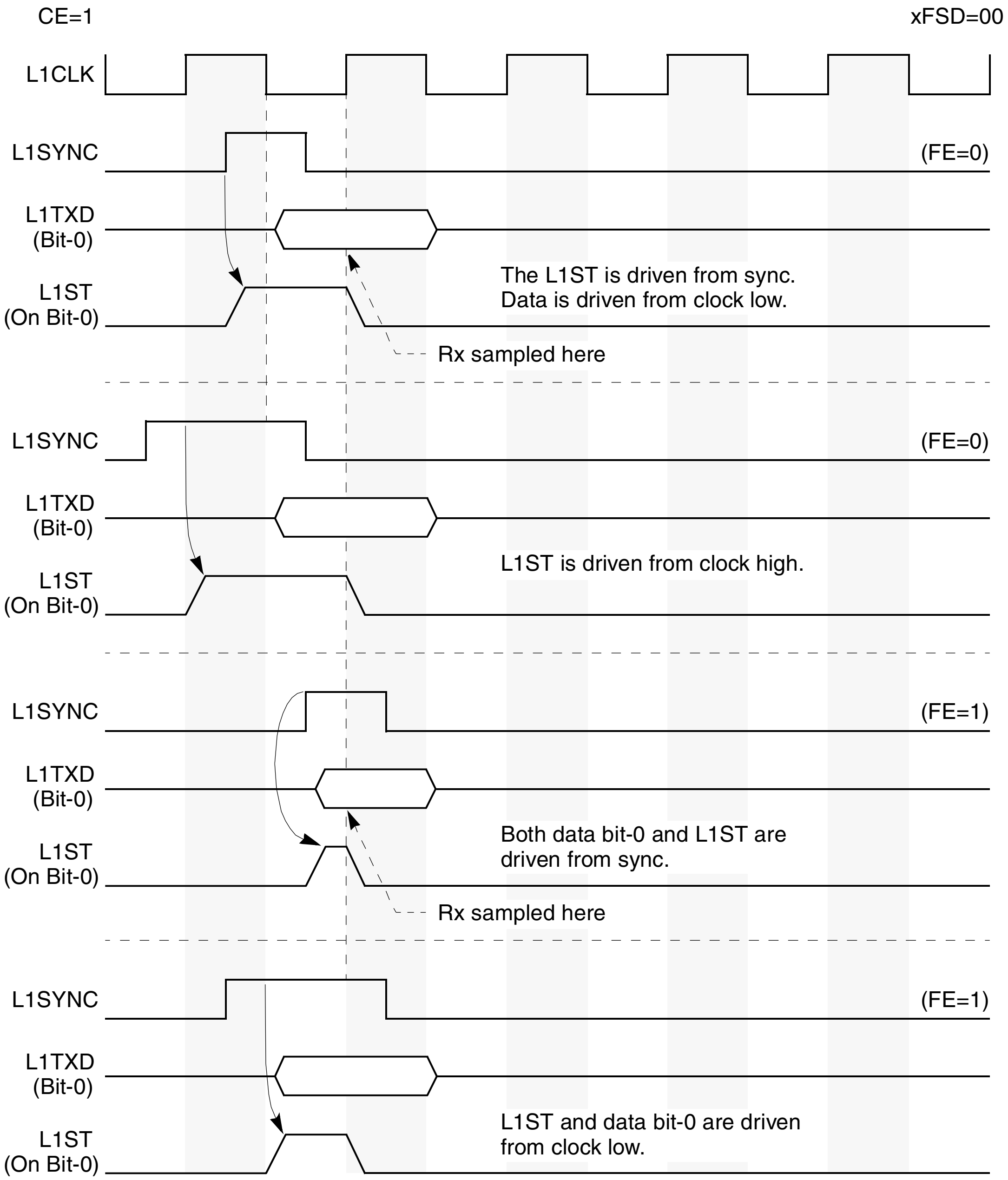


Figure 14-16. Falling Edge (FE) Effect when CE = 1 and *x*FSD = 00

Figure 14-17 shows the effects of changing FE when CE = 0 with no frame sync delay.

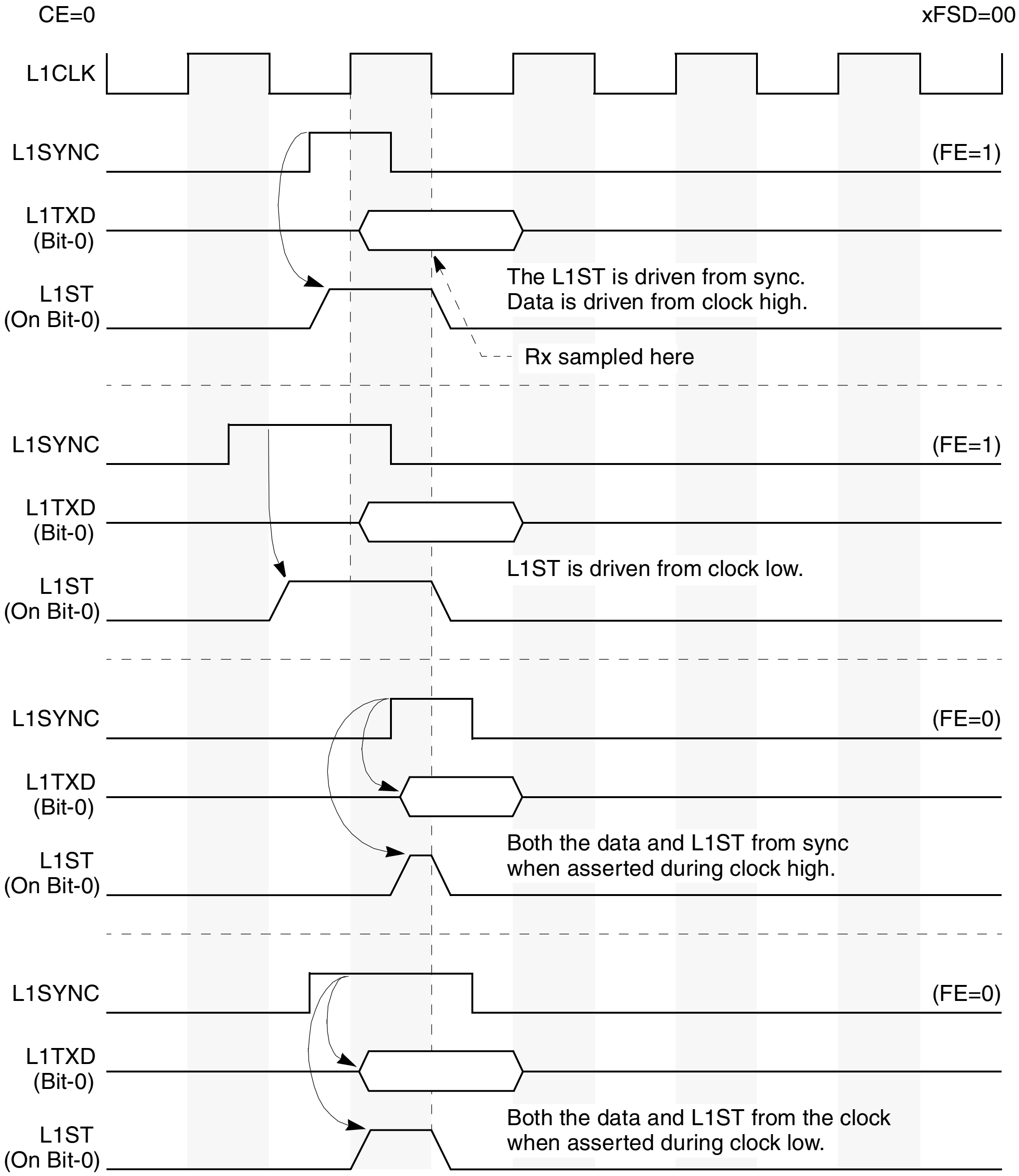

CE=0
xFSD=00
L1CLK
L1SYNC
(FE=1)
L1TXD
(Bit-0)
L1ST
(On Bit-0)
The L1ST is driven from sync.
Data is driven from clock high.
Rx sampled here
L1SYNC
(FE=1)
L1TXD
(Bit-0)
L1ST
(On Bit-0)
L1ST is driven from clock low.
L1SYNC
(FE=0)
L1TXD
(Bit-0)
L1ST
(On Bit-0)
Both the data and L1ST from sync
when asserted during clock high.
L1SYNC
(FE=0)
L1TXD
(Bit-0)
L1ST
(On Bit-0)
Both the data and L1ST from the clock
when asserted during clock low.

Figure 14-17. Falling Edge (FE) Effect when CE = 0 and *x*FSD = 00

14.5.3 SI2 RAM Shadow Address Register (SI2RSR)

The SI2 RAM shadow address register (SI2RSR), shown in Figure 14-18, defines the starting addresses of the shadow section in the SI2 RAM for each of the TDM channels.

	0	1	3	4	5	7	8	9	11	12	13	15
Field	—	SSADA		—	SSADB		—					
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x11B4E (SI2RSR)											

Figure 14-18. SI2 RAM Shadow Address Register (SI2RSR)

Table 14-5 describes SI2RSR fields.

Table 14-5. SI2RSR Field Descriptions

Bits	Name	Description
0,4, 8-12	—	Reserved. Should be cleared.
1–3,5–7	SSAD*x*	Starting bank address for the shadow RAM of TDM a or b. Defines the starting bank address of the shadow SI2 RAM section that belongs to the corresponding TDM channel. **Note:** As noted before, the SI2 RAM contain four banks of 64 entries for receive and four banks of 64 entries for transmit. In spite of the above, the starting bank address of each TDM can be programmed by the user in a granularity of 32 entries, but the user cannot put two different TDMs on the same bank. The user can put the shadow RAM section of the same TDM on the same bank. The last entry of a certain TDM frame is determined by the LST bit in the SI2 RAM entry. The user must set this bit within the entries of SI2 RAM shadow blocks for every TDM used. That means before the starting address of the next TDM.

14.5.4 SI Command Register (SI2CMDR)

The SI2 command register (SI2CMDR), shown in Figure 14-19, allows the user to dynamically program the SI2 RAM. When the user sets bits in SI2CMDR, the SI2 switches to the shadow SI2 RAM at the end of the current-route RAM programming frame. For more information about dynamic programming, see Section 14.4.5, “Static and Dynamic Routing.”

	0	1	2	3	4	5	6	7
Field	CSRRA	CSRTA	CSRRB	CSRTB	—	—	—	—
Reset	0000_0000							
R/W	R/W							
Addr	0x11B4A (SI2CMDR)							

Figure 14-19. SI Command Register (SI2CMDR)

Table 14-6 describes SI2CMDR fields.

Table 14-6. SI2CMDR Field Description

Bits	Name	Description
0, 2	CSRRx	Change shadow RAM for TDM a or b receiver. Set CSRRx causes the SI receiver to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The receiver shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The receiver shadow RAM is valid. The SI exchanges between the RAMs and take the new receive routing from the receiver shadow RAM. Cleared as soon as the switch has completed.
1, 3	CSRTx	Change shadow RAM for TDM a or b transmitter. Set CSRTx causes the SI transmitter to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The transmitter shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The transmitter shadow RAM is valid. The SI exchanges between the RAMs and take the new transmitter routing from the receiver shadow RAM. Cleared as soon as the switch has completed.
4 - 7	—	Reserved. Should be cleared.

14.5.5 SI Status Register (SI2STR)

The SI2 status register (SI2STR), shown in Figure 14-20, identifies the current-route RAM. SI2STR values are valid only when the corresponding SI2CMDR bit = 0.

	0	1	2	3	4	5	6	7
Field	CRORA	CROTA	CRORB	CROTB	—	—	—	—
Reset	0000_0000							
R/W	R							
Addr	0x11B4C (SI2STR)							

Figure 14-20. SI Status Register (SI2STR)

Table 14-7 describes SI2STR fields.

Table 14-7. SI2STR Field Descriptions

Bits	Name	Description
0, 2	CRORx	Current-route original receiver. Determines whether the current-route receiver RAM is the original or the shadow. 0 The current-route receiver RAM is the lower address area. 1 The current-route receiver RAM is the upper address area.
1, 3	CROTx	Current-route original transmitter. Determines whether the current-route transmitter RAM is the original or the shadow. 0 The current-route transmitter RAM is the lower address area. 1 The current-route transmitter RAM is the upper address area.
4 - 7	—	Reserved. Should be cleared.

14.6 Serial Interface IDL Interface Support

The IDL interface is a full-duplex ISDN interface used to connect a physical layer device to the PowerQUICC II. The PowerQUICC II supports both the basic and primary rate of the IDL bus. In the basic rate of IDL, data on three channels (B1, B2, and D) is transferred in a 20-bit frame, providing a full-duplex bandwidth of 160 Kbps. The PowerQUICC II is an IDL slave device that is clocked by the IDL bus master (physical layer device) and has separate receive and transmit sections. The PowerQUICC II has two TDMs, and it can support only two independent IDL buses (limited by the number of TDMs) using separate clocks and sync pulses. Figure 14-21 shows an application with two IDL buses.

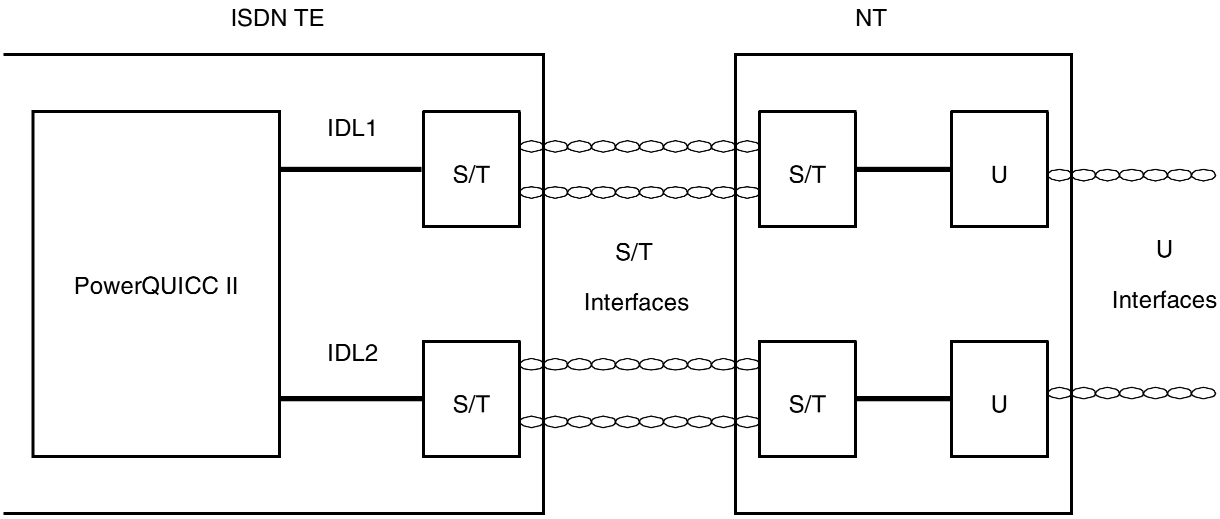


Figure 14-21. Dual IDL Bus Application Example

14.6.1 IDL Interface Example

An example of the IDL application is the ISDN terminal adaptor shown in Figure 14-22. In such an application, the IDL interface is used to connect the 2B+D channels between the PowerQUICC II, CODEC, and S/T transceiver. One of the PowerQUICC II’s SCCs is configured to HDLC mode to handle the D channel; another PowerQUICC II’s SCC is used to rate adapt the terminal data stream over the first B channel. That SCC is configured for HDLC mode if V.120 rate adoption is required. The second B channel is then routed to the CODEC as a digital voice channel, if preferred. The SPI is used to send initialization commands and periodically check status from the S/T transceiver. The SMC connected to the terminal is configured for UART.

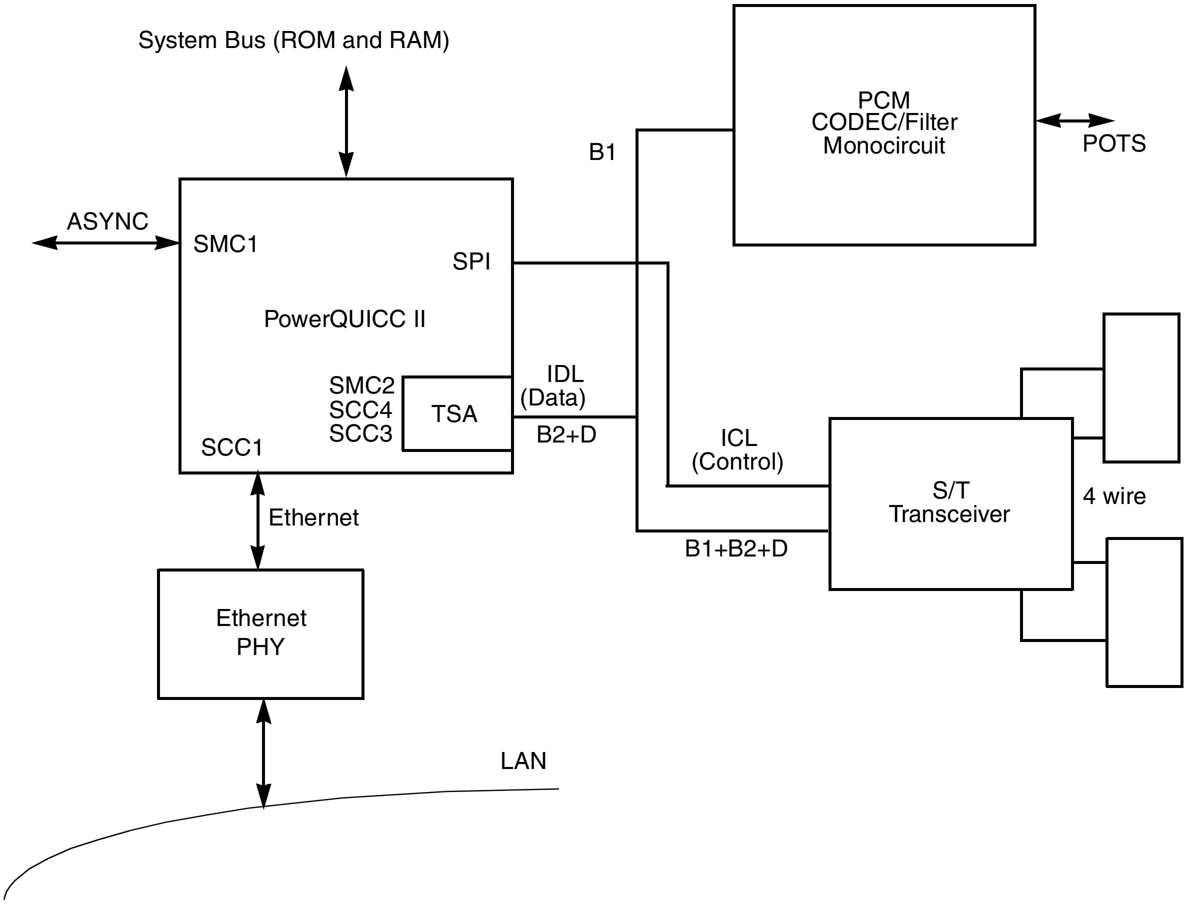


Figure 14-22. IDL Terminal Adaptor

The PowerQUICC II can identify and support each IDL channel or can output strobe lines for interfacing devices that do not support the IDL bus. The IDL signals for each transmit and receive channel are described in Table 14-8.

Table 14-8. IDL Signal Descriptions

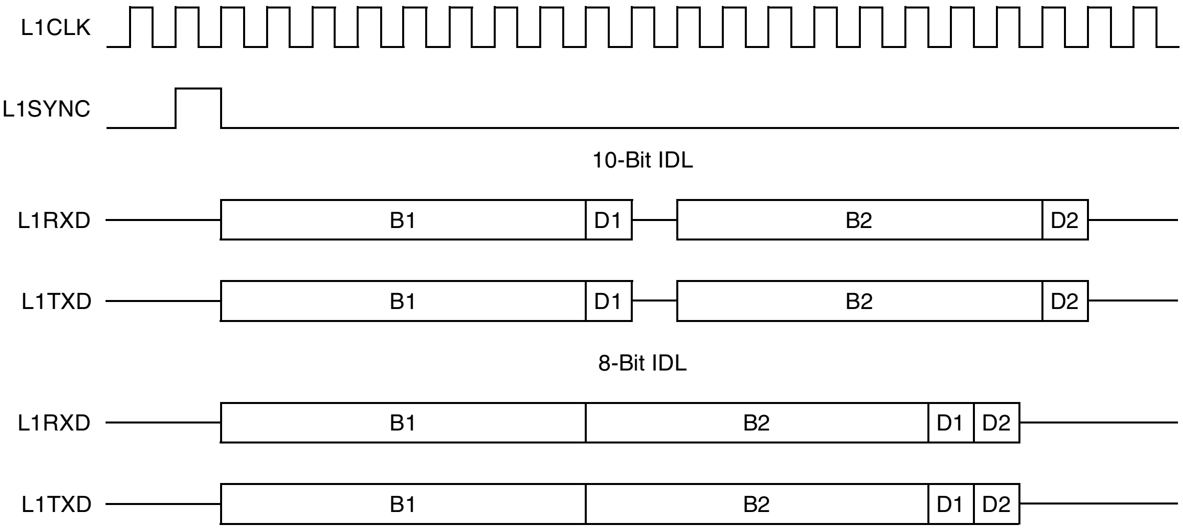
Signal	Description
L1RCLK*x*	IDL clock; input to the PowerQUICC II.
L1RSYNC*x*	IDL sync signal; input to the PowerQUICC II. This signal indicates that the clock periods following the pulse designate the IDL frame.
L1RXD*x*	IDL receive data; input to the PowerQUICC II. Valid only for the bits supported by the IDL; ignored for any other signals present.
L1TXD*x*	IDL transmit data; output from the PowerQUICC II. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
$\overline{\text{L1RQ}x}$	IDL request permission to transmit on the D channel; output from the PowerQUICC II on the $\overline{\text{L1RQ}x}$ pin.
L1GR*x*	IDL grant permission to transmit on the D channel; input to the PowerQUICC II on the L1TSYNCx pin.

Note: *x* = a and b for TDMa and TDMb.

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 14-23.



Notes:
1. Clocks are not to scale.
2. $\overline{\text{L1RQ}x}$ and L1GR*x* are not shown.

Figure 14-23. IDL Bus Signals

NOTE

Previous versions of Freescale IDL-defined bit functions called auxiliary (A) and maintenance (M) were removed from the IDL definition when it was concluded that the IDL control channel would be out-of-band. These functions were defined as a subset of the Freescale SPI format called serial control port (SCP). To implement the A and M bits as originally defined, program the TSA to access these bits and route them transparently to an SCC or SMC. Use the SPI to perform out-of-band signaling.

The PowerQUICC II supports all channels of the IDL bus in the basic rate. Each bit in the IDL frame can be routed to any SCC and SMC or can assert a strobe output for supporting an external device. The PowerQUICC II supports the request-grant method for contention detection on the D channel of the IDL basic rate and when the PowerQUICC II has data to transmit on the D channel, it asserts $\overline{\text{L1RQ}x}$. The physical layer device monitors the physical layer bus for activity on the D channel and indicates that the channel is free by asserting L1GRx. The PowerQUICC II samples the L1GRx signal when the IDL sync signal (L1RSYNCx) is asserted. If L1GRx is asserted, the PowerQUICC II sends the first zero of the opening flag in the first bit of the D channel. If a collision is detected on the D channel, the physical layer device negates L1GRx. The PowerQUICC II 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 PowerQUICC II supports up to four 8-bit channels in the frame, determined by the SI2 RAM programming. To support more channels, the user can route more than one channel to each SCC and the SCC treats it as one high-speed stream and store it in the same data buffers (appropriate only for transparent data). Additionally, the PowerQUICC II 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.

14.6.2 IDL Interface Programming

To program an IDL interface, first program SI2MR[GM*x*] to the IDL grant mode for that channel. If the receive and transmit sections interface to the same IDL bus, set SI2MR[CRT*x*] to internally connect the Rx clock and sync signals to the transmit section. Then, program the SI2 RAM used for the IDL channels to the preferred routing. See Section 14.4.4, “SI2 RAM Programming Example.”

Define the IDL frame structure by programming SI2MR[*x*FSD*x*] to have a 1-bit delay from frame sync to data, SI2MR[FE*x*] to sample on the falling edge, and SI2MR[CE*x*] to transmit on the rising edge of the clock. Program the parallel I/O open-drain register so that L1TXD*x* is three-stated when inactive; see Section 37.2.1, “Port Open-Drain Registers (PODRx).” To support the D channel, program the appropriate CMXSCR[GR*x*] bit, as described in Section 15.4.4, “CMX SCC Clock Route Register (CMXSCR),” and program the SI2 RAM entry to route data to the chosen serial controller. The two definitions of IDL, 8 or 10 bits, are implemented by simply modifying the SI2 RAM programming. In both cases, L1GR*x* is sampled while L1TSYNC*x* is asserted and transferred to the D-channel SCC as a grant indication.

For example, based on the same 10-bit format as in Section 14.4.4, “SI2 RAM Programming Example,” implement an IDL bus using SCC1, SCC3, and SMC1 connected to TDMa2 as follows:

1. Program both the Tx and Rx sections of the SI2 RAM as in Table 14-9.

Table 14-9. SI2 RAM Entries for an IDL Interface

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8-bit SCC1
1	0	0000	0001	000	0	0	1-bit SCC1
2	0	0000	0000	000	0	0	1-bit no support
3	0	0000	0101	011	0	0	4-bit SMC1

Table 14-9. SI2 RAM Entries for an IDL Interface (continued)

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
4	0	0000	0101	011	0	0	4-bit SMC1
5	0	1000	0001	000	0	1	1-bit SCC1 strobe1

2. CMXSI2CR = 0x00. TDMA receive clock is CLK13.
3. CMXSMR = 0x80. SMC1 is connected to the TSA.
4. CMXSCR = 0xC000_4000. SCC1 and SCC3 are connected to the TSA. SCC1 supports the grant mechanism because it handles the D channel.
5. SI2AMR = 0x0145. TDMA grant mode is used with 1-bit frame sync delay in Tx and Rx and common receive-transmit mode.
6. Set PPARD[20–22], PPARC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
7. Set PSORD[20–22], PSORC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
8. Clear PDIRD[22]. Configures L1TXDA(in/out).
9. Set PODRD[22]. Configures L1TXDA to an open-drain output.
10. Set PPARC[19]. Configures L1RCLKA.
11. Clear PDIRC[19] Configures L1RCLKA.
12. Clear PSORC[19]. Configures L1RCLKA.
13. Set PPARC[1]. Configures $\overline{\text{L1RQa}}$.
14. Set PSORC[1]. Configures $\overline{\text{L1RQa}}$.
15. Set PDIRC[1]. Configures $\overline{\text{L1RQa}}$.
16. Set PPARC[8]. Configures L1ST1.
17. Set PSORC[8]. Configures L1ST1.
18. Set PDIRC[8]. Configures L1ST1.
19. SI2CMDR is not used.
20. SI2STR does not need to be read.
21. Configure the SCC1 for HDLC operation (to handle the LAPD protocol of the D channel), and configure SCC3 and SMC1 as preferred.
22. SI2GMR = 0x01. Enable TDM A (one static TDM).
23. Enable SCC1, SCC3 and SMC1.

14.7 Serial Interface GCI Support

The PowerQUICC II fully supports the normal mode of the GCI, also known as the ISDN-oriented modular revision 2.2 (IOM-2), and the SCIT. The PowerQUICC II also supports the D-channel access control in S/T interface terminals using the command/indication (C/I) channel

The GCI bus consists of four lines—two data lines, a clock, and a frame synchronization line. Usually, an 8-KHz frame structure defines the various channels within the 256-Kbps data rate. The PowerQUICC II supports two (limited by the number of SMCs and TDMs) independent GCI buses, each with independent receive and transmit sections. The interface can also be used in a multiplexed frame structure on which up to eight physical layer devices multiplex their GCI channels. In this mode, the data rate would be 2,048 Kbps.

In the GCI bus, the clock rate is twice the data rate. The SI divides the input clock by two to produce the data clock. The PowerQUICC II also has data strobe lines and the 1× data rate clock L1CLKOx output pins. These signals are used for interfacing devices to GCI that do not support the GCI bus. Table 14-10 describes GCI signals for each transmit and receive channel.

Table 14-10. GCI Signals

Signal	Description
L1RSYNC*x*	Used as a GCI sync signal; input to the PowerQUICC II. This signal indicates that the clock periods following the pulse designate the GCI frame.
L1RCLK*x*	Used as a GCI clock; input to the PowerQUICC II. The L1RCLK*x* signal frequency is twice the data clock.
L1RXD*x*	Used as a GCI receive data; input to the PowerQUICC II.
L1TXD*x*	Used as a GCI transmit data; open-drain output. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
L1CLKO*x*	Optional signal; output from the PowerQUICC II. This 1× clock output is used to clock devices that do not interface directly to the GCI. If the double-speed clock is used, (DSC*x* bit is set in the SI2MR), this output is the L1RCLK*x* divided by 2; otherwise, it is simply a 1× output of the L1RCLKx signal.

Note: *x* = a and b for TDMa and TDMb.

The GCI bus signals are shown in Figure 14-24.

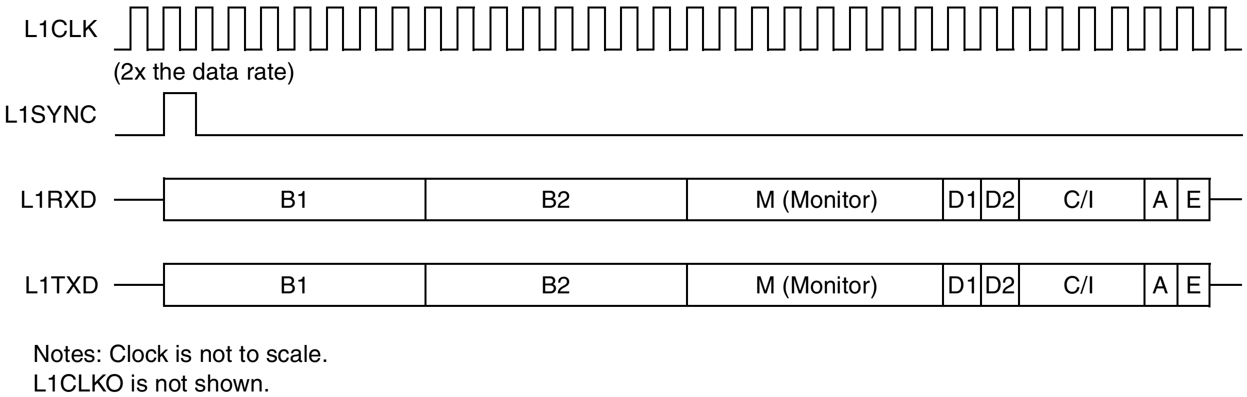


Figure 14-24. GCI Bus Signals

In addition to the 144-Kbps ISDN 2B+D channels, the GCI provides five channels for maintenance and control functions:

- B1 is a 64-Kbps bearer channel.
- B2 is a 64-Kbps bearer channel.

- M is a 64-Kbps monitor channel.
- D is a 16-Kbps signaling channel.
- C/I is a 48-Kbps C/I channel (includes A and E bits).

The M channel is used to transfer data between layer 1 devices and the control unit (the CPU); the C/I channel is used to control activation/deactivation procedures or to switch test loops by the control unit. The M and C/I channels of the GCI bus should be routed to SMC1 or SMC2, which have modes to support the channel protocols. The PowerQUICC II can support any channel of the GCI bus in the primary rate by modifying SI2 RAM programming.

The GCI supports the CCITT I.460 recommendation as a method for data rate adaptation since it can access each bit of the GCI separately. The current-route RAM specifies which bits are supported by the interface and which serial controller support them. The receiver only receives the bits that are enabled by the SI2 RAM and the transmitter only transmits the bits that are enabled by the SI2 RAM and does not drive L1TXDx. Otherwise, L1TXDx is an open-drain output and should be pulled high externally.

The PowerQUICC II supports contention detection on the D channel of the SCIT bus. When the PowerQUICC II has data to transmit on the D channel, it checks a SCIT bus bit that is marked with a special route code (usually, bit 4 of C/I channel 2). The physical layer device monitors the physical layer bus for activity on the D channel and indicates on this bit that the channel is free. If a collision is detected on the D channel, the physical layer device sets bit 4 of C/I channel 2 to logic high. The PowerQUICC II then aborts its transmission and retransmits the frame when this bit is set again. This procedure is automatically handled for the first two buffers of a frame.

14.7.1 SI GCI Activation/Deactivation Procedure

In the deactivated state, the clock pulse is disabled and the data line is at a logic one. The layer 1 device activates the PowerQUICC II by enabling the clock pulses and by an indication in the channel 0 C/I channel. The PowerQUICC II reports to the core (through a maskable interrupt) that a valid indication is in the SMC RxBD.

When the core activates the line, the data output of L1TXDn is programmed to zero by setting SI2GMR[STZx]. Code 0 (command timing TIM) is transmitted on channel 0 C/I channel to the layer 1 device until STZx is reset. The physical layer device resumes the clock pulses and gives an indication in the channel 0 C/I channel. The core should reset STZx to enable data output.

14.7.2 Serial Interface GCI Programming

The following sections describe serial interface GCI programming.

14.7.2.1 Normal Mode GCI Programming

The user can program and configure the channels used for the GCI bus interface. First, the SI2MR register to the GCI/SCIT mode for that channel must be programmed, using the DSCx, FEx, CEx, and RFSDx bits. This mode defines the sync pulse to GCI sync for framing and data clock as one-half the input clock rate. The user can program more than one channel to interface to the GCI bus. Also, if the receive and transmit section are used for interfacing the same GCI bus, the user internally connects the receive clock and sync signals to the SI2 RAM transmit section, using the CRTx bits. The user should then define the GCI frame routing and strobe select using the SI2 RAM.

When the receive and transmit section uses the same clock and sync signals, these sections should be programmed to the same configuration. Also, the L1TXDx pin in the I/O register should be programmed to be an open-drain output. To support the monitor and the C/I channels in GCI, those channels should be routed to one of the SMCs. To support the D channel when there is no possibility of collision, the user should clear the SI2MR[GR*x*] bit corresponding to the SCC that supports the D channel.

14.7.2.2 SCIT Programming

For interfacing the GCI/SCIT bus, SI2MR must be programmed to the GCI/SCIT mode. The SI2 RAM is programmed to support a 96-bit frame length and the frame sync is programmed to the GCI sync pulse. Generally, the SCIT bus supports the D channel access collision mechanism. For this purpose, the user should program the CRT*x* bits so the receive and transmit sections use the same clock and sync signals and program the GR*x* bits to transfer the D channel grant to the SCC that supports this channel. The received (grant) bit should be marked by programming the channel select bits of the SI2 RAM to 0b0111 for an internal assertion of a strobe on this bit. This bit is sampled by the SI and transferred to the D-channel SCC as the grant. The bit is generally bit 4 of the C/I in channel 2 of the GCI, but any other bit can be selected using the SI2 RAM.

For example, assuming that SCC1 is connected to the D channel, SCC3 to the B1 channel, and SMC2 to the B2 channel, SMC1 is used to handle the C/I channels, and the D-channel grant is on bit 4 of the C/I on SCIT channel 2, the initialization sequence is as follows:

1. Program both the Tx and Rx sections of the SI2 RAM as in Table 14-11 beginning at addresses 0 and 1024, respectively.

Table 14-11. SI2 RAM Entries for a GCI Interface (SCIT Mode)

Entry Number	SI2 RAM Entry						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0000	0011	000	1	0	8-bit SCC3
1	0	0000	0110	000	1	0	8-bit SMC2
2	0	0000	0101	000	1	0	8-bit SMC1
3	0	0000	0001	001	0	0	2-bit SCC1
4	0	0000	0101	101	0	0	6-bit SMC1
5	0	0000	0000	110	1	0	Skip 7 bytes
6	0	0000	0000	001	0	0	Skip 2 bits
7	0	0000	0111	000	0	1	D grant bit

2. SI2AMR = 0x00c0. TDMa is used in double speed clock and common Rx/Tx modes. SCIT mode is used in this example.

NOTE

If SCIT mode is not used, delete the last three entries of the SI2 RAM, divide one entry into two and set the LST bit in the new last entry.

3. CMXSMR = 0x88. SMC1 and SMC2 are connected to the TSA.
4. CMXSCR = 0xC000_4000. SCC3 and SCC1 are connected to the TSA. SCC1 supports the grant mechanism since it is on the D channel.
5. CMXSI2CR = 0x00. TDMa uses CLK13.
6. Set PPARD[20–22], PPARC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
7. Set PSORD[20–22], PSORC[9]. Configures L1TXDA, L1RXDA, L1TSYNCA, and L1RSYNCA.
8. Clear PDIRD[22]. Configures L1TXDA(inout).
9. Set PODRD[22]. Configures L1TXDA to an open-drain output.
10. Set PPARC[19]. Configures L1RCLKA.
11. Clear PDIRC[19] Configures L1RCLKA.
12. Clear PSORC[19]. Configures L1RCLKA.
13. Set PPARC[1]. Configures $\overline{\text{L1RQa}}$.
14. Set PSORC[1]. Configures $\overline{\text{L1RQa}}$.
15. Set PDIRC[1]. Configures $\overline{\text{L1RQa}}$.
16. If the 1x GCI data clock is required, set PPARC bit 0 and PDIRC bit 0 and set PSORC 0, which configures L1CLKOa as an output.
17. Configure SCC1 for HDLC operation (to handle the LAPD protocol of the D channel). Configure SMC1 for SCIT operation and configure SCC3 and SMC2 as preferred.
18. SI2GMR = 0x11. Enable TDMa (one static TDM), STZ for TDMa.
19. SI2CMDR is not used.
20. SI2STR does not need to be read.
21. Enable the SCC1, SCC3, SMC1 and SMC2.

Chapter 15
CPM Multiplexing

The CPM multiplexing (CMX) logic connects the physical layer—UTOPIA, MII/RMII, modem lines, TDM lines and proprietary serial lines—to the FCCs, SCCs, SMCs, and USB. The CMX features the following two modes:

- NMSI mode—The CMX allows all serial devices to be connected to their own set of individual pins. Each serial device that connects to the external world in this way is said to connect to a non-multiplexed serial interface (NMSI). In the NMSI configuration, the CMX provides a flexible clocking assignment for each FCC, SCC, SMC, and USB from a bank of external clock pins and/or internal BRGs.
- TDM mode—The CMX performs the connection of the serial devices to the SI for using the time-slot assigner (TSA). This allows any combination of FCCs, SCCs, and SMCs to multiplex data on any of the TDM channels. The CMX connects the serial device only to the TSA in the SI. The actual multiplexing of the TDM is made by programming the SI RAM. In TDM mode, all other pins used in NMSI mode are available for other purposes.

NOTE

> The USB uses only the NMSI mode and is mutually exclusive with SCC3 in NMSI mode; it is not legal to enable both peripherals in NMSI mode at the same time; The USB and SCC3 in TDM mode can be enabled at the same time.

Figure 15-1 shows a block diagram of the CMX.

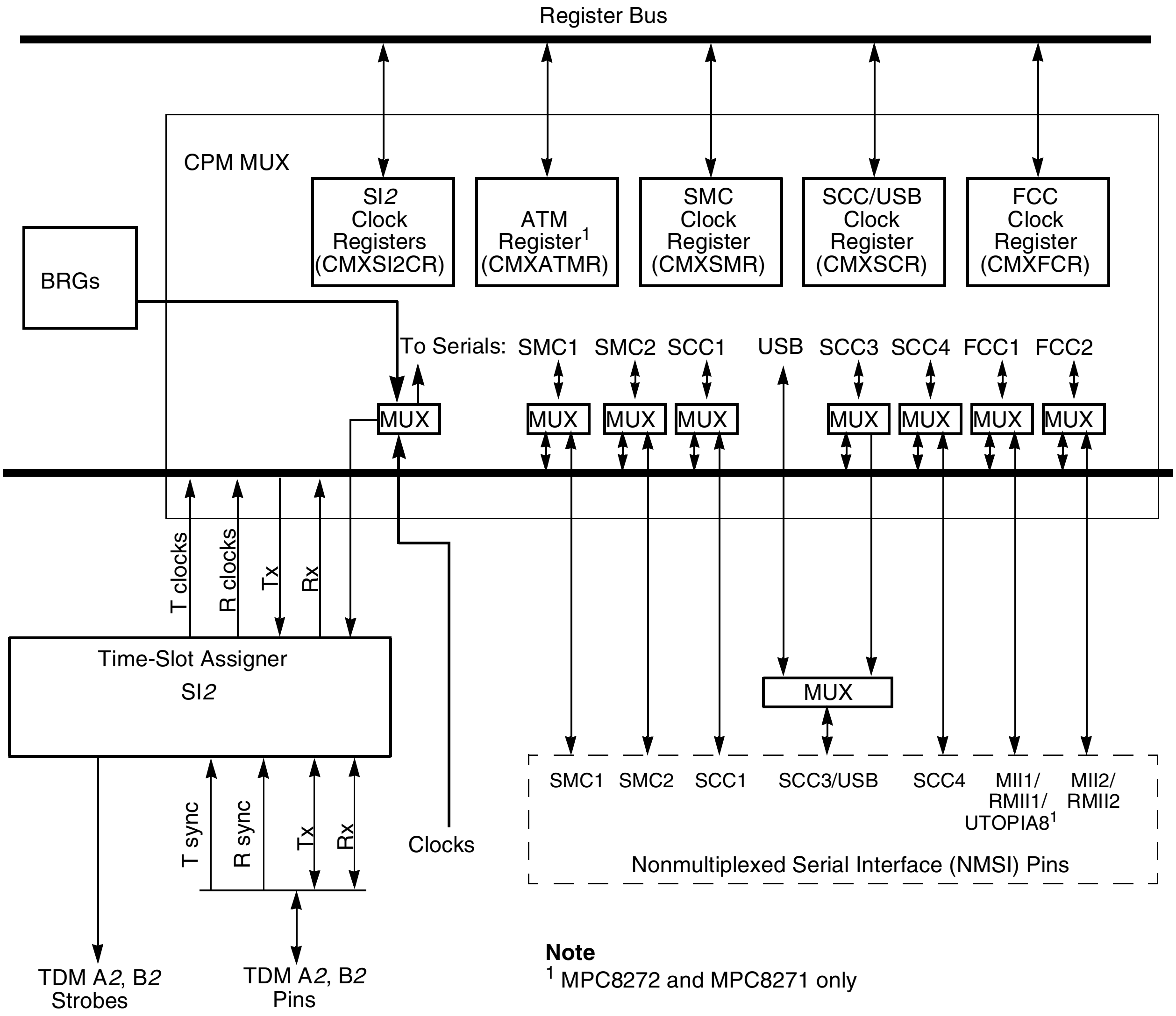

Register Bus
CPM MUX
SI2 Clock Registers (CMXSI2CR)
ATM Register[1] (CMXATMR)
SMC Clock Register (CMXSMR)
SCC/USB Clock Register (CMXSCR)
FCC Clock Register (CMXFCR)
BRGs
To Serials: SMC1 SMC2 SCC1 USB SCC3 SCC4 FCC1 FCC2
MUX
T clocks
R clocks
Tx
Rx
Time-Slot Assigner
SI2
T sync
R sync
Tx
Rx
Clocks
TDM A2, B2 Strobes
TDM A2, B2 Pins
SMC1 SMC2 SCC1 SCC3/USB SCC4 MII1/ RMII1/ UTOPIA8[1] MII2/ RMII2
Nonmultiplexed Serial Interface (NMSI) Pins
Note
[1] MPC8272 and MPC8271 only

Figure 15-1. CPM Multiplexing Logic (CMX) Block Diagram

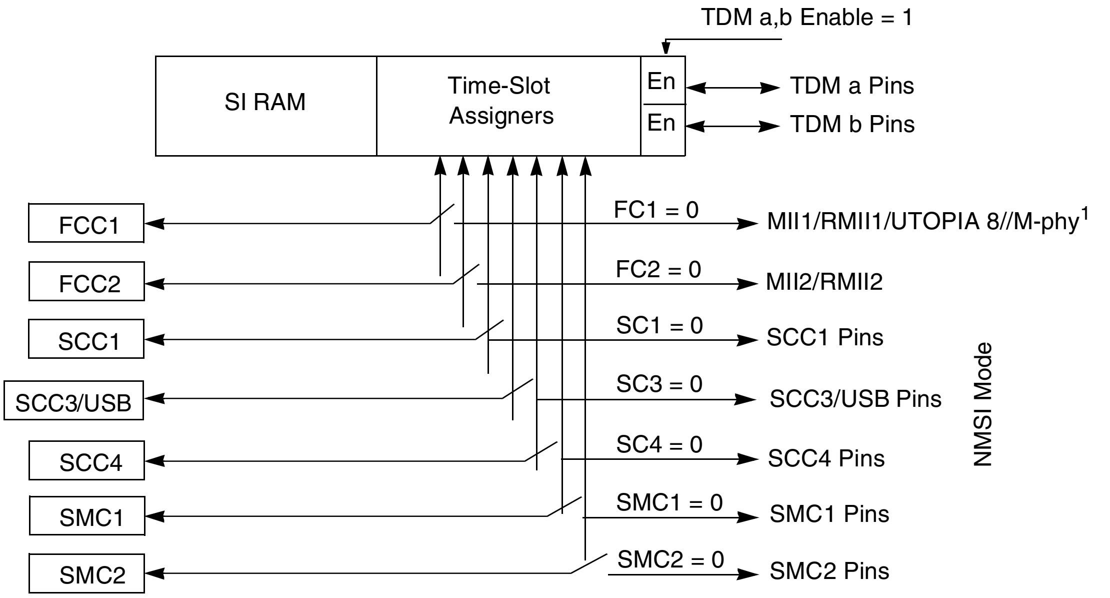
15.1 Features

The NMSI mode supports the following:

- Each FCC, SCC, and SMC can be programmed independently to work with a serial device’s own set of pins in a non-multiplexed manner.
- Each FCC can be connected to its own MII/RMII (media-independent interface).
- FCC1 can also be connected to an 8-bit ATM UTOPIA level-2 interface (MPC8272 only).
- Each SCC can have its own set of modem control pins.
- Each SMC can have its own set of four pins.
- Each FCC, SCC, SMC and the USB can be driven from a bank of fourteen clock pins or a bank of eight BRGs.

15.2 Enabling Connections to TSA or NMSI

Each serial device can be independently enabled to connect to the TSA or to dedicated external pins, as shown in Figure 15-2. Each FCC can be connected to a dedicated MII/RMII or to the two TDMs. FCC1 can also be connected to an 8-bit UTOPIA level-2 interface (MPC8272 only). Each SCC or SMC can be connected to the two TDMs or to its own set of pins. Once connections are made to the TSA, the exact routing decisions are made in the SI RAM.



Note
[1] MPC8272 and MPC8271only.

Figure 15-2. Enabling Connections to the TSA

15.3 NMSI Configuration

The CMX supports an NMSI mode for each of the FCCs, SCCs, and SMCs. Each of these serial devices is connected independently either to the NMSI or to the TSA using the clock route registers. The user should note, however, that NMSI pins are multiplexed with other functions at the parallel I/O lines. Therefore, if a combination of TDM and NMSI channels are used, consult the pinout to determine which FCC, SCC, and SMC to connect and where to connect them.

The clocks provided to the FCCs, SCCs, SMCs and USB are derived from a bank of 8 internal BRGs and 14 external CLK pins; see Figure 15-3. There are two main advantages to the bank-of-clocks approach. First, a serial device is not forced to choose a serial device clock from a predefined pin or BRG; this allows a flexible pinout-mapping strategy. Second, a group of serial receivers and transmitters that needs the same clock rate can share the same pin. This configuration leaves additional pins for other functions and minimizes potential skew between multiple clock sources.

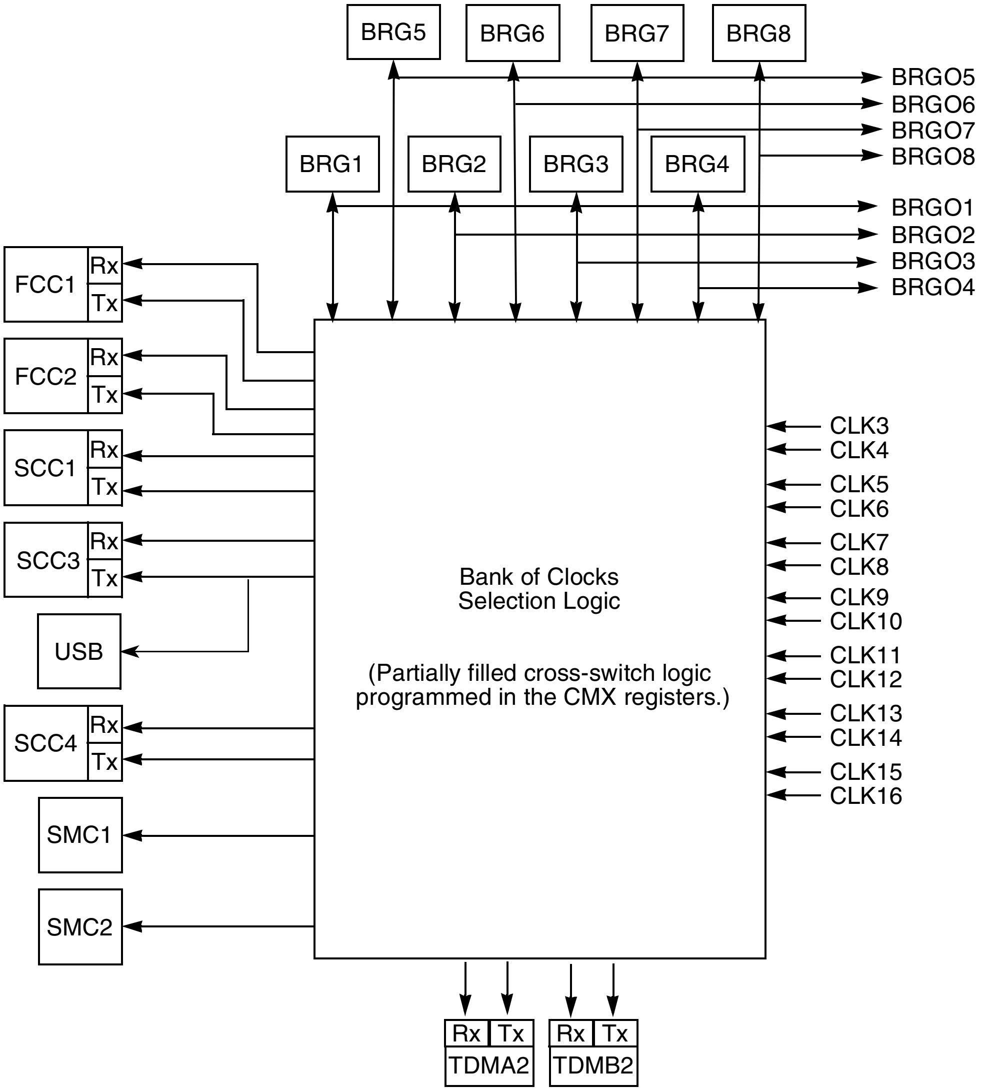


Figure 15-3. Bank of Clocks

The eight BRGs also make their clocks available to external logic, regardless of whether the BRGs are being used by a serial device. Notice that the BRG outputs are multiplexed with other functions; thus, all BRGO*x* pins may not always be available. Chapter 37, “Parallel I/O Ports,” shows the function multiplexing.

The following two restrictions apply in the bank-of-clocks mapping:

- Only four of the fourteen sources can be connected to any given FCC, SCC receiver or transmitter or to the USB.
- The SMC transmitter and receiver share the same clock source when connected to the NMSI.

Table 15-1 shows the clock source options for the serial controllers and TDM channels.

Table 15-1. Clock Source Options

Clock	CLK														BRG							
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
SCC1 Rx	V	V							V	V					V	V	V	V				
SCC1 Tx	V	V							V	V					V	V	V	V				
SCC3 Rx			V	V	V	V									V	V	V	V				
SCC3 Tx/USB			V	V	V	V									V	V	V	V				
SCC4 Rx			V	V	V	V									V	V	V	V				
SCC4 Tx			V	V	V	V									V	V	V	V				
FCC1 Rx							V	V	V	V									V	V	V	V
FCC1 Tx							V	V	V	V									V	V	V	V
FCC2 Rx											V	V	V	V					V	V	V	V
FCC2 Tx											V	V	V	V					V	V	V	V
TDMA2 Rx			V								V											
TDMA2 Tx				V								V										
TDMB2 Rx	V						V															
TDMB2 Tx		V						V														
SMC1 Rx					V		V								V						V	
SMC1 Tx					V		V								V						V	
SMC2 Rx		V											V			V						V
SMC2 Tx		V											V			V						V

NOTE

After a clock source is selected, the clock is given an internal name. For the FCCs and SCCs, the names are RCLK*x* and TCLK*x*; for SMCs, the name is simply SMCLK*x*; for USB the name is USBCLK. These internal names specify the clocks sent to the FCCs, SCCs, SMCs or USB. The internal names for the FCCs, SCCs or SMCs are used only in NMSI. These names do not correspond to any MPC8272 pins.

15.4 CMX Registers

The following sections describe the CMX registers.

15.4.1 CMX Utopia Register (CMXATMR) (MPC8272 only)

The CMX Utopia register (CMXATMR), shown in Figure 15-4, defines the connection of a BRG to FCC1 when an internal rate feature is used.

	0 7	8 9	10 15
Field	—	F1IRB	—
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x11B0E		

Figure 15-4. CMX Utopia Register (CMXATMR)

Table 15-2 describes CMXATMR fields.

Table 15-2. CMXATMR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–9	F1IRB	FCC1 internal rate BRG selection. Selects the BRG to be connected to FCC1 for internal rate operation. Used by the ATM controller 00 FCC1 internal rate clock is BRG5 01 FCC1 internal rate clock is BRG6 10 FCC1 internal rate clock is BRG7 11 FCC1 internal rate clock is BRG8
10–15	—	Reserved, should be cleared.

15.4.2 CMX SI2 Clock Route Register (CMXSI2CR)

The CMX SI2 clock route register (CMXSI2CR), seen in Figure 15-5, defines the connection of SI2 to the clock sources that can be input from the bank of clocks.

	0	1	2	3	4	5	6	7
Field	RTA2CS	RTB2CS	—	—	TTA2CS	TTB2CS	—	—
Reset	0000_0000							
R/W	R/W							
Addr	0x11B02							

Figure 15-5. CMX SI2 Clock Route Register (CMXSI2CR)

Table 15-3 describes CMXSI2CR fields.

Table 15-3. CMXSI2CR Field Descriptions

Bits	Name	Description
0	RTA2CS	Receive TDM A2 clock source 0 TDM A2 receive clock is CLK13 1 TDM A2 receive clock is CLK5
1	RTB2CS	Receive TDM B2 clock source 0 TDM B2 receive clock is CLK3 1 TDM B2 receive clock is CLK9
2–3	—	Reserved, should be cleared.

Table 15-3. CMXSI2CR Field Descriptions (continued)

Bits	Name	Description
4	TTA2CS	Transmit TDM A2 clock source 0 TDM A2 transmit clock is CLK14 1 TDM A2 transmit clock is CLK6
5	TTB2CS	Transmit TDM B2 clock source 0 TDM B2 transmit clock is CLK4 1 TDM B2 transmit clock is CLK10
6–7	—	Reserved, should be cleared.

15.4.3 CMX FCC Clock Route Register (CMXFCR)

The CMX FCC clock route register (CMXFCR), shown in Figure 15-6, defines the connection of the FCCs to the TSA and to the clock sources from the bank of clocks.

	0	1	2 4	5 7	8	9	10 12	13 15
Field	—	FC1	RF1CS	TF1CS	—	FC2	RF2CS	TF2CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x11B04							

	16 31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x11B06

Figure 15-6. CMX FCC Clock Route Register (CMXFCR)

Table 15-4 describes CMXFCR fields.

Table 15-4. CMXFCR Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1	FC1	Defines the FCC1 connection. 0 FCC1 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC1 is connected to the TSA of the SI. The NMSI2 pins are available for other purposes.

Table 15-4. CMXFCR Field Descriptions (continued)

Bits	Name	Description
2–4	RF1CS	Receive FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 receive clock is BRG5 001 FCC1 receive clock is BRG6 010 FCC1 receive clock is BRG7 011 FCC1 receive clock is BRG8 100 FCC1 receive clock is CLK9 101 FCC1 receive clock is CLK10 110 FCC1 receive clock is CLK11 111 FCC1 receive clock is CLK12
5–7	TF1CS	Transmit FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 transmit clock is BRG5 001 FCC1 transmit clock is BRG6 010 FCC1 transmit clock is BRG7 011 FCC1 transmit clock is BRG8 100 FCC1 transmit clock is CLK9 101 FCC1 transmit clock is CLK10 110 FCC1 transmit clock is CLK11 111 FCC1 transmit clock is CLK12
8	—	Reserved, should be cleared
9	FC2	Defines the FCC2 connection. 0 FCC2 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC2 is connected to the TSA of the SI. The NMSI pins are available for other purposes.
10–12	RF2CS	Receive FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1). 000 FCC2 receive clock is BRG5 001 FCC2 receive clock is BRG6 010 FCC2 receive clock is BRG7 011 FCC2 receive clock is BRG8 100 FCC2 receive clock is CLK13 101 FCC2 receive clock is CLK14 110 FCC2 receive clock is CLK15 111 FCC2 receive clock is CLK16
13–15	TF2CS	Transmit FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1). 000 FCC2 transmit clock is BRG5 001 FCC2 transmit clock is BRG6 010 FCC2 transmit clock is BRG7 011 FCC2 transmit clock is BRG8 100 FCC2 transmit clock is CLK13 101 FCC2 transmit clock is CLK14 110 FCC2 transmit clock is CLK15 111 FCC2 transmit clock is CLK16
16–31	—	Reserved, should be cleared

15.4.4 CMX SCC Clock Route Register (CMXSCR)

The CMX SCC clock route register (CMXSCR), shown in Figure 15-7, defines the connection of the SCCs to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

	0	1	2 4	5 7	8 15
Field	GR1	SC1	RS1CS	TS1CS	—
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x11B08				

	16	17	18 20	21 23	24	25	26 28	29 31
Field	GR3	SC3	RS3CS	TS3CS	GR4	SC4	RS4CS	TS4CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x11B0A							

Figure 15-7. CMX SCC Clock Route Register (CMXSCR)

Table 15-5 describes CMXSCR fields.

Table 15-5. CMXSCR Field Descriptions

Bits	Name	Description
0	GR1	Grant support of SCC1 0 SCC1 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC1 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
1	SC1	SCC1 connection 0 SCC1 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC1 is connected to TSA of the SI. The NMSI2 pins are available for other purposes.
2–4	RS1CS	Receive SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1). 000 SCC1 receive clock is BRG1 001 SCC1 receive clock is BRG2 010 SCC1 receive clock is BRG3 011 SCC1 receive clock is BRG4 100 SCC1 receive clock is CLK11 101 SCC1 receive clock is CLK12 110 SCC1 receive clock is CLK3 111 SCC1 receive clock is CLK4
5–7	TS1CS	Transmit SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1). 000 SCC1 transmit clock is BRG1 001 SCC1 transmit clock is BRG2 010 SCC1 transmit clock is BRG3 011 SCC1 transmit clock is BRG4 100 SCC1 transmit clock is CLK11 101 SCC1 transmit clock is CLK12 110 SCC1 transmit clock is CLK3 111 SCC1 transmit clock is CLK4

Table 15-5. CMXSCR Field Descriptions (continued)

Bits	Name	Description
8–15	—	Reserved, should be cleared
16	GR3	Grant support of SCC3 0 SCC3 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC3 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
17	SC3	SCC3 connection 0 SCC3 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC3 is connected to TSA of the SI. The NMSI2 pins are available for other purposes.
18–20	RS3CS	Receive SCC3 clock source (NMSI mode). Ignored if SCC3 is connected to the TSA (SC3 = 1). 000 SCC3 receive clock is BRG1 001 SCC3 receive clock is BRG2 010 SCC3 receive clock is BRG3 011 SCC3 receive clock is BRG4 100 SCC3 receive clock is CLK5 101 SCC3 receive clock is CLK6 110 SCC3 receive clock is CLK7 111 SCC3 receive clock is CLK8
21–23	TS3CS	Transmit SCC3 clock source (NMSI mode). Used as the USB clock when SCC3 is disabled or connected to the TSA (SC3 = 1). 000 SCC3 transmit/USB clock is BRG1 001 SCC3 transmit/USB clock is BRG2 010 SCC3 transmit/USB clock is BRG3 011 SCC3 transmit/USB clock is BRG4 100 SCC3 transmit/USB clock is CLK5 101 SCC3 transmit/USB clock is CLK6 110 SCC3 transmit/USB clock is CLK7 111 SCC3 transmit/USB clock is CLK8
24	GR4	Grant support of SCC4 0 SCC4 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC4 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
25	SC4	SCC4 connection 0 SCC4 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC4 is connected to TSA of the SI. The NMSI2 pins are available for other purposes.

Table 15-5. CMXSCR Field Descriptions (continued)

Bits	Name	Description
26–28	RS4CS	Receive SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1). 000 SCC4 receive clock is BRG1 001 SCC4 receive clock is BRG2 010 SCC4 receive clock is BRG3 011 SCC4 receive clock is BRG4 100 SCC4 receive clock is CLK5 101 SCC4 receive clock is CLK6 110 SCC4 receive clock is CLK7 111 SCC4 receive clock is CLK8
29–31	TS4CS	Transmit SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1). 000 SCC4 transmit clock is BRG1 001 SCC4 transmit clock is BRG2 010 SCC4 transmit clock is BRG3 011 SCC4 transmit clock is BRG4 100 SCC4 transmit clock is CLK5 101 SCC4 transmit clock is CLK6 110 SCC4 transmit clock is CLK7 111 SCC4 transmit clock is CLK8

15.4.5 CMX SMC Clock Route Register (CMXSMR)

The CMX SMC clock route register (CMXSMR), shown in Figure 15-8, defines the connection of the SMCs to the TSA and to the clock sources from the bank of clocks.

	0	1	2	3	4	5	6	7
Field	SMC1	—	SMC1CS		SMC2	—	SMC2CS	
Reset	0000_0000							
R/W	R/W							
Addr	0x11B0C							

Figure 15-8. CMX SMC Clock Route Register (CMXSMR)

Table 15-6 describes CMXSMR fields.

Table 15-6. CMXSMR Field Descriptions

Bits	Name	Description
0	SMC1	SMC1 connection 0 SMC1 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus SMCn pins is made in the parallel I/O control register. 1 SMC1 is connected to the TSA of the SI. The NMSI2 pins are available for other purposes.
1	—	Reserved, should be cleared.

Table 15-6. CMXSMR Field Descriptions (continued)

Bits	Name	Description
2–3	SMC1CS	SMC1 clock source (NMSI mode). SMC1 can take its clocks from one of the two BRGs or one of two pins from the bank of clocks. However, the SMC1 transmit and receive clocks must be the same when it is connected to the NMSI. 00 SMC1 transmit and receive clocks are BRG1 01 SMC1 transmit and receive clocks are BRG7 10 SMC1 transmit and receive clocks are CLK7 11 SMC1 transmit and receive clocks are CLK9
4	SMC2	SMC2 connection 0 SMC2 is not connected to the TSA and is either connected directly to the NMSI2 pins or is not used. The choice of general-purpose I/O port pins versus SMCn pins is made in the parallel I/O control register. 1 SMC2 is connected to the TSA of the SI. The NMSI2 pins are available for other purposes.
5	—	Reserved, should be cleared
6–7	SMC2CS	SMC2 clock source (NMSI mode). SMC2 can take its clocks from one of the eight BRGs or one of eight pins from the bank of clocks. However, the SMC2 transmit and receive clocks must be the same when it is connected to the NMSI. 00 SMC2 transmit and receive clocks are BRG2 01 SMC2 transmit and receive clocks are BRG8 10 SMC2 transmit and receive clocks are CLK4 11 SMC2 transmit and receive clocks are CLK15

Chapter 16
Baud-Rate Generators (BRGs)

The CPM contains eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs, SCCs, and SMCs. The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they can be routed to the controllers. In addition, the output of a BRG can be routed to a pin to be used externally. The following is a list of the main features of BRGs:

- Eight independent BRGs
- On-the-fly changes allowed
- Each BRG can be routed to one or more FCCs, SCCs, or SMCs.
- A 16x divider option allows slow baud rates at high system frequencies.
- BRGs 1, 3, and 4 contain an autobaud support option (for SCCs 1, 3, and 4).
- Each BRG output can be routed to a pin (BRGO*n*).

Figure 16-1 shows a BRG.

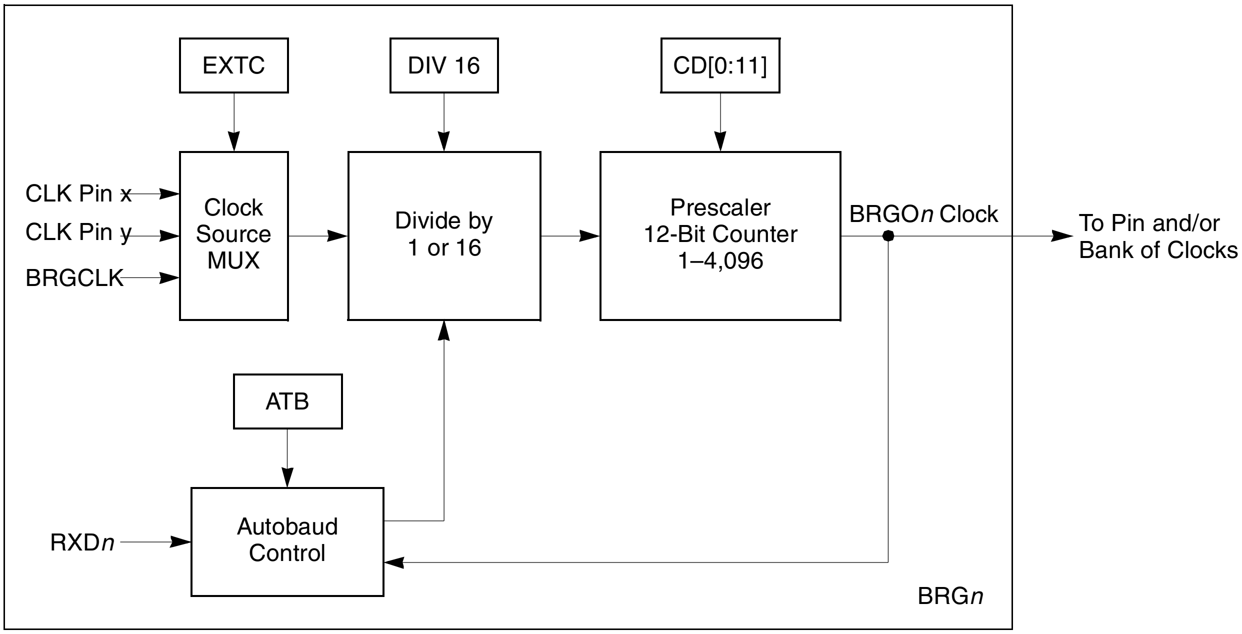


Figure 16-1. Baud-Rate Generator (BRG) Block Diagram

Each BRG clock source can be BRGCLK, or a choice of two external clocks (selected in BRGC*x*[EXTC]). The BRGCLK is an internal signal generated in the MPC8272 clock synthesizer specifically for the BRGs, the SPI, and the I^{2}C internal BRG. Alternatively, external clock pins can be configured as clock sources. The external source option allows flexible baud-rate frequency generation, independent of the system frequency. Additionally, the external source option allows a single external frequency to be the source for multiple BRGs. The external source signals are not synchronized internally before being used by the BRG.

The BRG provides a divide-by-16 option (BRGC*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; however, two changes should not occur within two source clock periods.

The prescaler output is sent internally to the bank of clocks and can also be output externally on BRGO*n* through the parallel I/O ports. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the falling edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. Additionally, the output of the BRG can be sent to the autobaud control block.

16.1 BRG Configuration Registers 1–8 (BRGC*x*)

The BRG configuration registers (BRGC*x*) are shown in Figure 16-2. A reset disables the BRG and drives the BRGO output clock high. The BRGC can be written at any time with no need to disable the SCCs or external devices that are connected to BRGO. Configuration changes occur at the end of the next BRG clock cycle (no spikes occur on the BRGO output clock). BRGC can be changed on-the-fly; however, two changes should not occur within a time equal to two source clock periods.

<table>
<tr><td></td><td>0</td><td>13</td><td>14</td><td>15</td></tr>
<tr><td>Field</td><td colspan="2">—</td><td>RST</td><td>EN</td></tr>
<tr><td>Reset</td><td colspan="4">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="4">R/W</td></tr>
<tr><td>Addr</td><td colspan="4">0x119F0 (BRGC1), 0x119F4 (BRGC2), 0x119F8 (BRGC3), 0x119FC (BRGC4),
0x115F0 (BRGC5), 0x115F4 (BRGC6), 0x1115F8 (BRGC7), 0x115FC (BRGC8)</td></tr>
</table>

<table>
<tr><td></td><td>16</td><td>17</td><td>18</td><td>19</td><td>30</td><td>31</td></tr>
<tr><td>Field</td><td colspan="2">EXTC</td><td>ATB</td><td colspan="2">CD</td><td>DIV16</td></tr>
<tr><td>Reset</td><td colspan="6">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="6">R/W</td></tr>
<tr><td>Addr</td><td colspan="6">0x119F22 (BRGC1), 0x119F6 (BRGC2), 0x119FA (BRGC3), 0x119FE (BRGC4),
0x115F2 (BRGC5), 0x115F6 (BRGC6), 0x115FA (BRGC7), 0x115FE (BRGC8)</td></tr>
</table>

Figure 16-2. Baud-Rate Generator Configuration Registers (BRGC*x*)

Table 16-1 describes the BRGC*x* fields.

Table 16-1. 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.
16–17	EXTC	External clock source. Selects the BRG input clock. See Table 16-2.. 00 The BRG input clock comes from the BRGCLK (internal clock generated from the CPM clock); see Section 10.4, "System Clock Control Register (SCCR)." 01 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK3 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK9 pin 10 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK5 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK15 pin 11 Reserved
18	ATB	Autobaud. Selects autobaud operation of the BRG on the corresponding RXD. ATB must remain zero until the SCC receives the 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. **Note:** Because there is no SCC2, ATB cannot function on BRG2. 0 Normal operation of the BRG. 1 When RXD goes low, the BRG determines the length of the start bit and synchronizes the BRG to the actual baud rate.
19–30	CD	Clock divider. CD presets an internal 12-bit counter that is decremented at the DIV16 output rate. When the counter reaches zero, it is reloaded with CD. CD = 0xFFF produces the minimum clock rate for BGRO (divide by 4,096); CD = 0x000 produces the maximum rate (divide by 1). When dividing by an odd number, the counter ensures a 50% duty cycle by asserting the terminal count once on clock low and next on clock high. The terminal count signals counter expiration and toggles the clock. See Section 16.3, "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 16.3, "UART Baud Rate Examples." 0 Divide by 1. 1 Divide by 16.

Table 16-2 shows the possible external clock sources for the BRGs.

Table 16-2. BRG External Clock Source Options

BRG	CLK																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BRG1			V		V															
BRG2			V		V															
BRG3									V						V					
BRG4									V						V					
BRG5			V		V															
BRG6			V		V															
BRG7									V						V					
BRG8									V						V					

16.2 Autobaud Operation on a UART

During the autobaud process, a UART deduces the baud rate of its received character stream by examining the received pattern and its timing. A built-in autobaud control function automatically measures the length of a start bit and modifies the baud rate accordingly.

If the autobaud bit BRGC*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 begins counting the start bit length. During this time, the BRG output clock toggles for 16 BRG clock cycles at the BRG source clock rate and stops with BRGO*n* in the low state.

When RXD*n* goes high again, the autobaud control block rewrites BRGC*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 SCC event register to report that the autobaud controller rewrote BRGC*x*. The interrupt handler can then adjust BRGC*x*[CD, DIV16] (see Table 16-3) for accuracy before the first character is fully received, ensuring that the UART recognizes all characters.

After a full character is received, the software can verify that the character matches a predefined value (such as ‘a’ or ‘A’). Software should then check for other characters (such as ‘t’ or ‘T’) and program the preferred parity mode in the UART’s protocol-specific mode register (PSMR).

Note that the SCC associated with this BRG must be programmed to UART mode and select the 16× option for TDCR and RDCR in the general SCC mode register low. Input frequencies such as 1.8432, 3.68, 7.36, and 14.72 MHz should be used. The SCC performing the autobaud function must be connected to that SCC’s BRG; that is, SCC3 must be clocked by BRG3, and so on.

Also, to detect an autobaud lock and generate an interrupt, the SCC must receive three full Rx clocks from the BRG before the autobaud process begins. To do this, first clear BRGC*x*[ATB] and enable the BRG Rx clock to the highest frequency. Then, immediately before the autobaud process starts (after device initialization), set BRGC*x*[ATB].

16.3 UART Baud Rate Examples

For synchronous communication using the internal BRG, the BRGO must not exceed the BRG input clock divided by 2. Therefore, with a BRG input clock of 66 MHz (generated using an external clock source: refer to BRGCx[EXTC]), the maximum BRGO rate is 33MHz. Program the UART to 16x oversampling when using the SCC as a UART. Rates of 8x and 32x are also available. Assuming 16x oversampling is chosen in the UART, the maximum data rate is 66 MHz ÷ 16 = 4.125 Mbps. Keeping the above in mind, use the following formula to calculate the bit rate based on a particular BRG configuration for a UART:

$$\text{Async Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \bullet (\text{Clock Divider} + 1) \bullet (\text{Sampling Rate})}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \bullet (\text{BRGCx[CD]} + 1) \bullet (\text{GSMRx\_L[xDCR]})}$$

Table 16-3 lists typical bit rates of asynchronous communication. Note that here the internal clock rate is assumed to be 16x the baud rate; that is, GSMR*x*_L[TDCR] = GSMR*x*_L[RDCR] = 0b10.

Table 16-3. 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	38,551
57,600	0	71	57,292
115,200	0	35	114,583
460,000	0	8	458,333

For synchronous communication, the internal clock is identical to the baud-rate output. To get the preferred rate, select the system clock according to the following:

$$\text{Sync Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \bullet (\text{Clock Divider} + 1)}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \bullet (\text{BRGCx[CD]} + 1)}$$

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.

Chapter 17
Timers

The CPM includes four identical 16-bit general-purpose timers or two 32-bit timers. Each general-purpose timer consists of a timer mode register (TMR), a timer capture register (TCR), a timer counter (TCN), a timer reference register (TRR), a timer event register (TER), and a timer global configuration register (TGCR). The TMRs contain the prescaler values programmed by the user.

Figure 17-1 shows the timer block diagram.

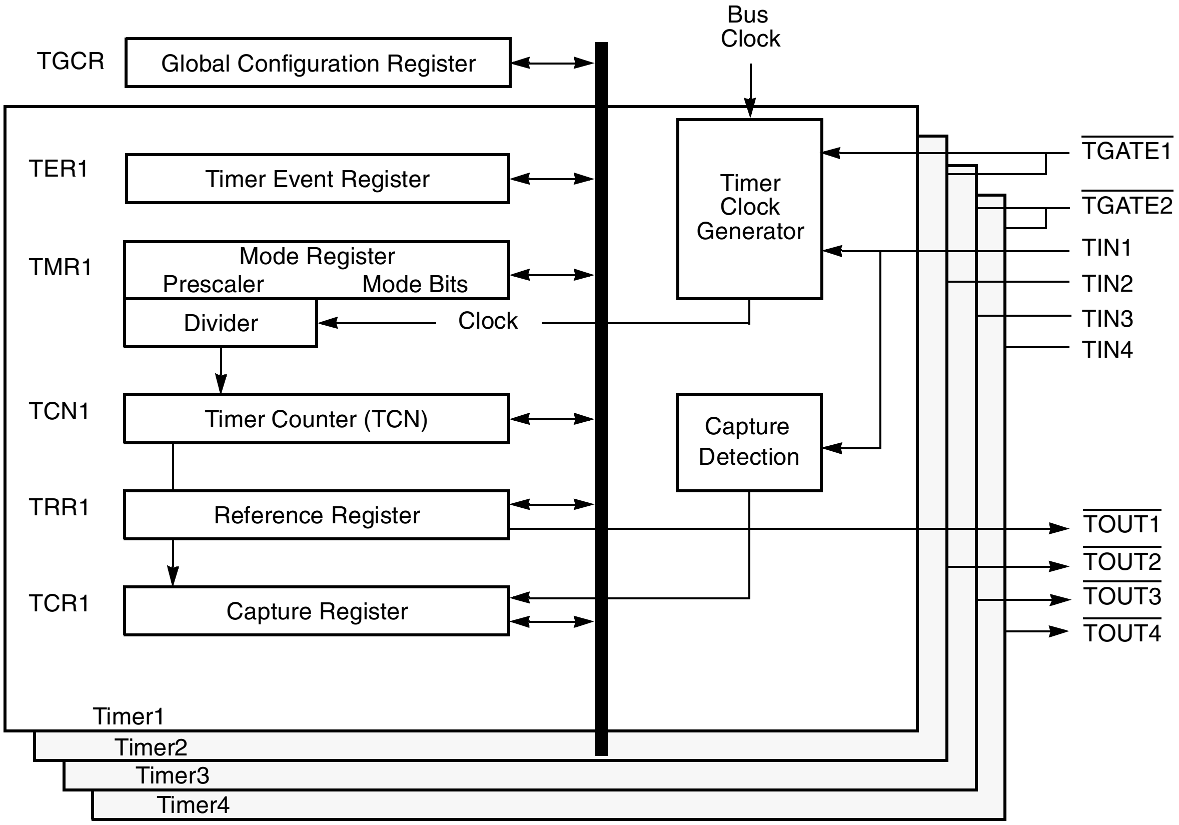


Figure 17-1. Timer Block Diagram

Pin assignments for TIN*x*, $\overline{\text{TGATE}\textit{x}}$, and $\overline{\text{TOUT}\textit{x}}$ are described in Section 37.5, “Ports Tables.”

17.1 Features

The key features of the timer include the following:

- The maximum input clock is the bus clock
- Maximum period of 4 seconds (at 66 MHz)

- 16-nanosecond resolution (at 66 MHz)
- Programmable sources for the clock input
- Input capture capability
- Output compare with programmable mode for the output pin
- Two timers cascade internally or externally to form a 32-bit timer
- Free run and restart modes
- Functional compatibility with timers on the MC68360 and MPC860

17.2 General-Purpose Timer Units

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

- The bus clock (CLKIN)
- The bus clock divided by 16 (CLKIN/16)
- The corresponding TIN*x*, programmed in the parallel port registers

The bus clock is generated in the clock synthesizer and defaults to the bus frequency. However, the bus clock has the option of being divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the bus 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*x* to be the clock source. TIN*x* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding TMR[ICLK] bits. The prescaler is programmed to divide the clock input by values from 1 to 256 and the output of the prescaler is used as an input to the 16-bit counter. The best resolution of the timer is one clock cycle (16 ns at 66 MHz). The maximum period (when the reference value is all 1s) is 268,435,456 cycles (4 seconds at 66 MHz).

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding TMR selects each mode. Upon reaching the reference value, the corresponding TER bit is set and an interrupt is issued if TMR[ORI] = 1. The timers can output a signal on the timer outputs ($\overline{\text{TOUT1}}$–$\overline{\text{TOUT4}}$) when the reference value is reached (selected by the corresponding TMR[OM]). This signal can be an active-low pulse or a toggle of the current output. The output can also be connected internally to the input of another timer, resulting in a 32-bit timer.

In addition, each timer has a 16-bit TCR used to latch the value of the counter when a defined transition of TIN1, TIN2, TIN3, or TIN4 is sensed by the corresponding input capture edge detector. The type of transition triggering the capture is selected by the corresponding TMR[CE] bits. Upon a capture or reference event, the corresponding TER bit is set and a maskable interrupt request is issued to the interrupt controller. The timers may be gated/restarted by an external gate signal. There are two gate signals—$\overline{\text{TGATE1}}$, which controls timer 1 and/or 2, and $\overline{\text{TGATE2}}$, which controls timer 3 and/or 4. Normal gate mode enables the count on a falling edge of $\overline{\text{TGATE}x}$ and disables the count on the rising edge of $\overline{\text{TGATE}x}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}x}$.

The restart gate mode performs the same function as normal mode, except that it also resets the counter on the falling edge of $\overline{\text{TGATE}x}$. This mode has applications in pulse interval measurement and bus monitoring as follows:

- Pulse measurement—The restart gate mode can measure a low $\overline{\text{TGATE}x}$. The rising edge of $\overline{\text{TGATE}x}$ completes the measurement and if $\overline{\text{TGATE}x}$ is connected externally to TIN*x*, it causes the timer to capture the count value and generate a rising-edge interrupt.
- Bus monitoring—The restart gate mode can detect a signal that is abnormally stuck low. The bus signal should be connected to $\overline{\text{TGATE}x}$. The timer count is reset on the falling edge of the bus signal and if the bus signal does not go high again within the number of user-defined clocks, an interrupt can be generated.

The gate function is enabled in the TMR; the gate operating mode is selected in the TGCR.

NOTE

> $\overline{\text{TGATE}x}$ is internally synchronized to the bus clock. If $\overline{\text{TGATE}x}$ meets the asynchronous input setup time, the counter begins counting after one bus clock when working with the internal clock.

17.2.1 Cascaded Mode

In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter. Timer 1 may be internally cascaded to timer 2, and timer 3 can be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user can select two 16-bit timers or one 32-bit timer. TGCR is used to put the timers into cascaded mode, as shown in Figure 17-2.

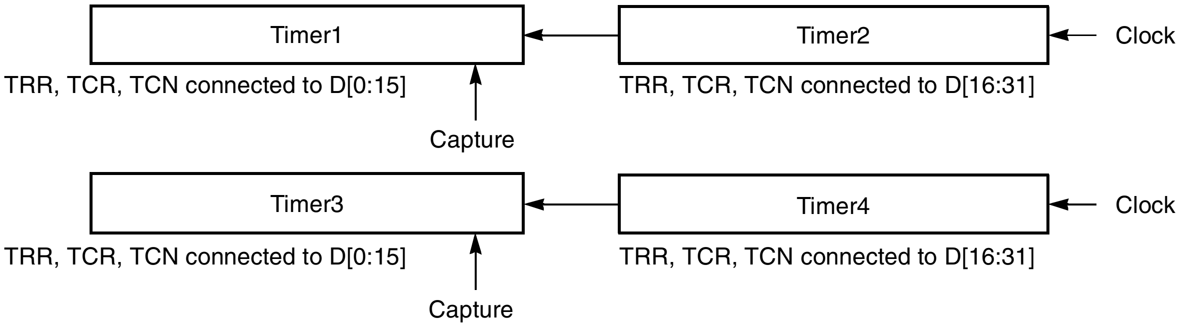

Timer1
Timer2
Clock
TRR, TCR, TCN connected to D[0:15]
TRR, TCR, TCN connected to D[16:31]
Capture
Timer3
Timer4
Clock
TRR, TCR, TCN connected to D[0:15]
TRR, TCR, TCN connected to D[16:31]
Capture

Figure 17-2. Timer Cascaded Mode Block Diagram

If TGCR[CAS] = 1, the two timers function as a 32-bit timer with a 32-bit TRR, TCR, and TCN. In this case, TMR1 and/or TMR3 are ignored, and the modes are defined using TMR2 and/or TMR4. The capture is controlled from TIN2 or TIN4 and the interrupts are generated from TER2 or TER4. In cascaded mode, the combined TRR, TCR, and TCN must be referenced with 32-bit bus cycles.

17.2.2 Timer Global Configuration Registers (TGCR1 and TGCR2)

The timer global configuration registers (TGCR1 and TGCR2), shown in Figure 17-3 and Figure 17-4, contain configuration parameters used by the timers. These registers allow simultaneous starting and stopping of a pair of timers (1 and 2 or 3 and 4) if one bus cycle is used.

	0	1	2	3	4	5	6	7
Field	**CAS2**	—	**STP2**	**RST2**	**GM1**	—	**STP1**	**RST1**
Reset	0000_0000							
R/W	R/W							
Addr	0x10D80							

Figure 17-3. Timer Global Configuration Register 1 (TGCR1)

Table 17-1 describes TGCR1 fields.

Table 17-1. TGCR1 Field Descriptions

Bits	Name	Description
0	**CAS2**	Cascade timers. 0 Normal operation 1 Timers 1 and 2 cascade to form a 32-bit timer.
1	—	Reserved, should be cleared.
2	**STP 2**	Stop timer. 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	**RST2**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM1**	Gate mode for $\overline{\text{TGATE1}}$. This bit is valid only if the gate function is enabled in TMR1 or TMR2. 0 Restart gate mode. $\overline{\text{TGATE1}}$ is used to enable/disable count. A falling $\overline{\text{TGATE1}}$ enables and restarts the count and a rising edge of $\overline{\text{TGATE1}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP1**	Stop timer. 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	**RST1**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

The TGCR2 register is shown in Figure 17-4.

	0	1	2	3	4	5	6	7
Field	**CAS4**	—	**STP4**	**RST4**	**GM2**	—	**STP3**	**RST3**
Reset	0000_0000							
R/W	R/W							
Addr	0x10D84							

Figure 17-4. Timer Global Configuration Register 2 (TGCR2)

Table 17-2 describes TGCR2 fields.

Table 17-2. TGCR2 Field Descriptions

Bit	Name	Description
0	**CAS4**	Cascade timers. 0 Normal operation 1 Timers 3 and 4 cascades to form a 32-bit timer.
1	—	Reserved, should be cleared.
2	**STP 4**	Stop timer. 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	**RST4**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM2**	Gate mode for $\overline{\text{TGATE2}}$. This bit is valid only if the gate function is enabled in TMR3 or TMR4. 0 Restart gate mode. $\overline{\text{TGATE2}}$ is used to enable/disable the count. The falling edge of $\overline{\text{TGATE2}}$ enables and restarts the count and the rising edge of $\overline{\text{TGATE2}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE2}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP3**	Stop timer. 0 Normal operation 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, however it is possible to read the values while the clock is stopped. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	**RST3**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

17.2.3 Timer Mode Registers (TMR1–TMR4)

The four timer mode registers (TMR1–TMR4) are shown in Figure 17-5.

Erratic behavior may occur if TGCR1 and TGCR2 are not initialized before the TMRs. Only TGCR[RST] can be modified at any time.

<table>
<tr><th></th><th>0</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2">PS</td><td colspan="2">CE</td><td>OM</td><td>ORI</td><td>FRR</td><td colspan="2">ICLK</td><td>GE</td></tr>
<tr><td>Reset</td><td colspan="10">0000_0000_0000_0000</td></tr>
<tr><td>R/W</td><td colspan="10">R/W</td></tr>
<tr><td>Addr</td><td colspan="10">0x10D90 (TMR1); 0x10D92 (TMR2); 0x10DA0 (TMR3); 0x10DA2 (TMR4)</td></tr>
</table>

Figure 17-5. Timer Mode Registers (TMR1–TMR4)

Table 17-3 describes TMR1–TMR4 register fields.

Table 17-3. TMR1–TMR4 Field Descriptions

Bits	Name	Description
0–7	**PS**	Prescaler value. The prescaler is programmed to divide the clock input by values from 1 to 256. The value 00000000 divides the clock by 1 and 11111111 divides the clock by 256.
8–9	**CE**	Capture edge and enable interrupt. 00 Disable interrupt on capture event; capture function is disabled. 01 Capture on rising TIN*x* edge only and enable interrupt on capture event 10 Capture on falling TIN*x* edge only and enable interrupt on capture event 11 Capture on any TIN*x* edge and enable interrupt on capture event
10	**OM**	Output mode 0 Active-low pulse on $\overline{\mathrm{TOUT}x}$ for one timer input clock cycle as defined by the ICLK bits. Thus, $\overline{\mathrm{TOUT}x}$ may be low for one bus clock period, one bus clock/16 period, or one TIN*x* clock cycle period. $\overline{\mathrm{TOUT}x}$ changes occur on the rising edge of the bus clock. 1 Toggle $\overline{\mathrm{TOUT}x}$. $\overline{\mathrm{TOUT}x}$ changes occur on the rising edge of the bus clock.
11	**ORI**	Output reference interrupt enable. 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt upon reaching the reference value.
12	**FRR**	Free run/restart. 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.
13–14	**ICLK**	Input clock source for the timer. 00 Internally cascaded input. For TMR1, the timer 1 input is the output of timer 2. For TMR3, the timer 3 input is the output of timer 4. For TMR2 and TMR4, this selection means no input clock is provided to the timer. 01 Internal bus clock. 10 Internal bus clock divided by 16. 11 Corresponding TIN*x*: TIN1, TIN2, TIN3, or TIN4 (falling edge).
15	**GE**	Gate enable. 0 $\overline{\mathrm{TGATE}x}$ is ignored. 1 $\overline{\mathrm{TGATE}x}$ is used to control the timer.

17.2.4 Timer Reference Registers (TRR1–TRR4)

Each timer reference register (TRR1–TRR4), shown in Figure 17-6, contains the timeout's reference value. The reference value is not reached until TCN*x* increments to equal the timeout reference value.

<table>
<tr><th></th><th>0</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2"><b>Timeout reference value</b></td></tr>
<tr><td>Reset</td><td colspan="2">0xFFFF</td></tr>
<tr><td>R/W</td><td colspan="2">R/W</td></tr>
<tr><td>Addr</td><td colspan="2">0x10D94 (TRR1), 0x10D96 (TRR2), 0x10DA4 (TRR3), 0x10DA6 (TRR4)</td></tr>
</table>

Figure 17-6. Timer Reference Registers (TRR1–TRR4)

17.2.5 Timer Capture Registers (TCR1–TCR4)

Each timer capture register (TCR1–TCR4), shown in Figure 17-7, is used to latch the value of the counter according to TMR*x*[CE].

<table>
<tr><th></th><th>0</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2">Latched counter value</td></tr>
<tr><td>Reset</td><td colspan="2">0x0000</td></tr>
<tr><td>R/W</td><td colspan="2">R/W</td></tr>
<tr><td>Addr</td><td colspan="2">0x10D98 (TCR1), 0x10D9A (TCR2), 0x10DA8 (TCR3), 0x10DAA (TCR4)</td></tr>
</table>

Figure 17-7. Timer Capture Registers (TCR1–TCR4)

17.2.6 Timer Counters (TCN1–TCN4)

Each timer counter register (TCN1–TCN4), shown in Figure 17-8, is an up-counter. A read cycle to TCN*x* yields the current value of the timer but does not affect the counting operation. A write cycle to TCN*x* sets the register to the written value, thus causing its corresponding prescaler, TMR*x*[PS], to be reset.

<table>
<tr><th></th><th>0</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2"><b>Up counter</b></td></tr>
<tr><td>Reset</td><td colspan="2">0x0000</td></tr>
<tr><td>R/W</td><td colspan="2">R/W</td></tr>
<tr><td>Addr</td><td colspan="2">0x10D9C (TCN1), 0x10D9E (TCN2), 0x10DAC (TCN3), 0x10DAE (TCN4)</td></tr>
</table>

Figure 17-8. Timer Counter Registers (TCN1–TCN4)

Note that the counter registers may not be updated correctly if a write is made while the timer is not running. Use TRR*x* to define the preferred count value.

17.2.7 Timer Event Registers (TER1–TER4)

Each timer event register (TER*x*), shown in Figure 17-9, reports events recognized by the timers. When an output reference event is recognized, the timer sets TER*x*[REF] regardless of the corresponding TMR*x*[ORI]. The capture event is set only if it is enabled by TMR*x*[CE]. TER1–TER4 can be read at any time.

Writing ones clears event bits; writing zeros has no effect. Both event bits must be cleared before the timer negates the interrupt.

	0	13	14	15
Field	—		REF	CAP
Reset	0x0000			
Addr	0x10DB0 (TER1); 0x10DB2 (TER2); 0x10DB4 (TER3); 0x10DB6 (TER4)			

Figure 17-9. Timer Event Registers (TER1–TER4)

Table 17-4 describes TER fields.

Table 17-4. TER Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	REF	Output reference event. The counter has reached the TRR value. TMR[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Capture event. The counter value has been latched into the TCR. TMR[CE] is used to enable generation of this event.

Chapter 18
SDMA Channels and IDMA Emulation

The PowerQUICC II has two physical serial DMA (SDMA) channels. The CP implements two dedicated virtual SDMA channels for each FCC, SCC, SMC, USB, SPI, and I^{2}C—one for each transmitter and receiver. An additional two virtual SDMA channels are assigned to the programmable independent DMA (IDMA) channels.

Figure 18-1 shows data flow paths. Data from the peripheral controllers can be routed to external RAM using the 60x bus (path 1).

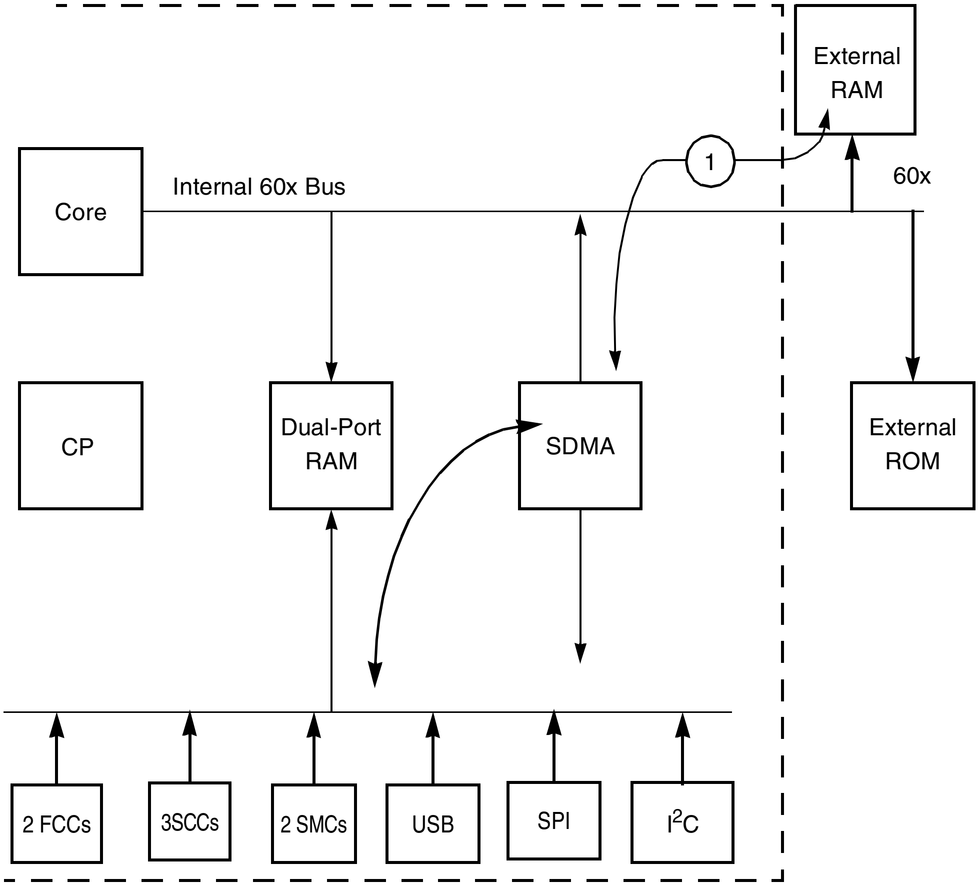


Figure 18-1. SDMA Data Paths

On a path 1 access, the SDMA channel must acquire the external system bus.

The SDMA channel can be assigned big-endian (Freescale) or little-endian format for accessing buffer data. These features are programmed in the receive and transmit registers associated with the FCCs, SCCs, SMCs, USB, SPI, and I^{2}C.

If a 60x bus error occurs on a CP-related access by the SDMA, the CP generates a unique interrupt in the SDMA status register (SDSR). The interrupt service routine then reads the appropriate DMA transfer error address register (PDTEA) to determine the address the bus error occurred on. The channel that caused the bus error is determined by reading the channel number from PDTEM. If an SDMA bus error occurs on a CP-related transaction, all CPM activity stops and the entire CPM must be reset in the CP command register (CPCR). See Section 18.2, “SDMA Registers.”

18.1 SDMA Bus Arbitration and Bus Transfers

On the PowerQUICC II, the core, PCI bridge, and SDMA can become external bus masters. (The relative priority of these masters is programmed by the user; see Section 4.3.2, “System Configuration and Protection Registers,” for programming bus arbitration.)Therefore, any SDMA channel can arbitrate for the bus against the other internal devices and any external devices present. Once an SDMA channel becomes system bus master, it remains bus master for one transaction (which can be a byte, half word, word, burst, or extended special burst) before releasing the bus. This feature, combined with the zero-clock arbitration overhead provided by the 60x bus, increases bus efficiency and lowers bus latency.

To minimize the latency associated with slower, character-oriented protocols, an SDMA writes each character to memory as it arrives without waiting for the next character, and always reads using 16-bit half-word transfers.

The SDMA can access the 60x bus either at the regular 60x transactions (single-beat accesses, four-beat bursts) or special two- and three-beat burst accesses. For a further description of this feature see Section 8.4.3.8, “Extended Transfer Mode.”

A transfer may take multiple bus transactions if the memory provides a less than 64-bit 60x port size. An SDMA uses back-to-back bus transactions for the entire transfer—4-word bursts, 64-bit reads, and 8-, 16-, 32-, or 64-bit writes—before relinquishing the bus. For example, a 64-bit word 60x-bus read from a 32-bit memory takes two consecutive SDMA bus transactions.

An SDMA can steal transactions with no arbitration overhead when the PowerQUICC II is bus master. Figure 18-2 shows an SDMA stealing a transaction from an internal bus master.

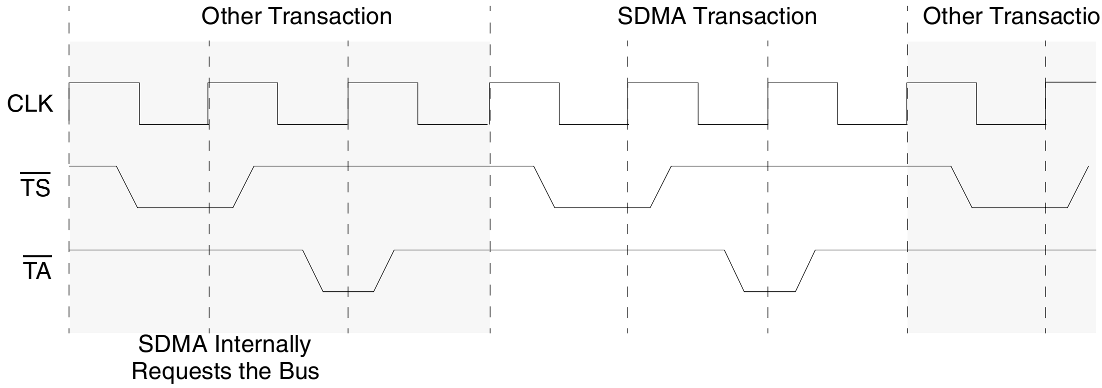


Figure 18-2. SDMA Bus Arbitration (Transaction Steal)

18.2 SDMA Registers

The only user-accessible registers associated with the SDMA are the SDMA address registers, read-only register used for diagnostics in case of an SDMA bus error, the SDMA status register and the SDMA mask register.

18.2.1 SDMA Status Register (SDSR)

The SDMA status register (SDSR), seen in Figure 18-3, reports bus error events recognized by the SDMA controller for all 26 SDMA channels and 4 IDMA channels. On recognition of a bus error on the 60x bus, the SDMA sets its corresponding SDSR bit. The 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.

	0	5	6	7
Field	—			SBER_P
Reset	0000_0000			
Addr	0x11018			

Figure 18-3. SDMA Status Register (SDSR)

Table 18-1 describes SDSR fields.

Table 18-1. SDSR Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	SBER_P	SDMA channel 60x bus error. Indicates that the SDMA channel on the 60x bus had terminated with an error during a read or write transaction. This bit is cleared writing a 1; writing a zero has no effect. The SDMA transfer error address is read from PDTEA. The channel number is read from PDTEM.

18.2.2 SDMA Mask Register (SDMR)

The SDMA mask register (SDMR) is an 8-bit read/write register with the same bit format as the SDMA status register. If an SDMR bit is 1, the corresponding interrupt in SDSR is enabled. If the bit is zero, the corresponding interrupt in the status register is masked. SDMR is cleared at reset. SDMR can be accessed at 0x1101C.

18.2.3 SDMA Transfer Error Address Register (PDTEA)

There is one 32-bit, read-only SDMA address register. The PDTEA holds the system address accessed during an SDMA transfer error on the 60x bus. It is undefined at reset. PDTEA can be accessed at 0x10050.

18.2.4 SDMA Transfer Error MSNUM Register (PDTEM)

There is one SDMA transfer error MSNUM register (PDTEM). MSNUM[0–4] contains the sub-block code (SBC) used to identify the current peripheral controller accessing the bus. MSNUM[5] identifies which half of the controller is transferring (transmitter or receiver). The MSNUM of each transaction is held in these registers until the transaction is complete. It is undefined at reset. See Figure 18-4.

	0 1	2 7
Field	—	MSNUM
Reset	—	
R/W	R	
Addr	0x10054 (PDTEM)	

Figure 18-4. SDMA Transfer Error MSNUM Registers (PDTEM)

Table 18-2 describes PDTEM fields.

Table 18-2. PDTEM Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–6	MSNUM[2–6]	Bits 2–6 of MSNUM is the sub-block code of the current peripheral controller accessing the bus. See the SBC field description of the CPCR in Section 13.4.1, “CP Command Register (CPCR).”
7	MSNUM[7]	Bit 7 of MSNUM indicates which section of the peripheral controller is accessing the bus. 0 Transmit section 1 Receive section

18.3 IDMA Emulation

The CPM can be configured to provide general-purpose DMA functionality through the SDMA channel. Two general-purpose independent DMA (IDMA) channels are supported. In this special emulation mode, the user can specify any memory-to-memory or peripheral-to/from-memory transfers as if using dedicated DMA hardware.

The general-purpose IDMA channels can operate in different user-programmable data transfer modes. The IDMA can transfer data between any combination of memory and I/O. In addition, data may be transferred in either byte, half-word, word, double-word or burst quantities (note that IDMA cannot burst to or from the dual-port RAM) and the source and destination addresses may be odd or even. The most efficient packing algorithms are used in the IDMA transfers; however, anytime the IDMA has 0x10 or more bytes to transfer, it will burst. The single-address mode (fly-by mode) gives the highest performance, allowing data to be transferred between memory and a peripheral in a single bus transaction. The chip-select and wait-state generation logic on the PowerQUICC II can be used with the IDMA.

The bus bandwidth occupied by the IDMA can be programmed in the IDMA parameter RAM to achieve maximum system performance.

The IDMA supports two buffer handling modes—auto buffer and buffer chaining. The auto buffer mode allows blocks of data to be repeatedly moved from one location to another without user intervention. The buffer chaining mode allows a chain of blocks to be moved. The user specifies the data movement using BD tables like those used by other peripheral controllers. The BD tables reside in the dual-port RAM.

Each IDMA has three signals (DREQ*x*, $\overline{\text{DACK}x}$ and $\overline{\text{DONE}x}$) for peripheral handshaking.

18.4 IDMA Features

The main IDMA features are as follows:

- Two independent, fully programmable DMA channels
- Dual- or single-address transfers with 32-bit address and 64-bit data capability
- Memory-to-memory, memory-to-peripheral, and peripheral-to-memory modes
- 4-Gbyte maximum block length for each buffer
- 32-bit address pointers that can be optionally incremented
- Two buffer handling modes—auto buffer and buffer chaining
- Interrupts are optionally generated for BD transfer completion, external $\overline{\text{DONE}}$ assertion, and STOP_IDMA command completion.
- Any channel is independently configurable for data transfer from any 60x, or PCI source to any 60x, or PCI destination
- Programmable byte-order conversion is supported independently for each DMA channel.
- Programmable 60x-bus bandwidth usage for system performance optimization is supported.

Peripheral to/from memory features include the following:

- External DREQ, $\overline{\text{DACK}}$, and $\overline{\text{DONE}}$ signals for each channel simplifies the peripheral interface for memory-to/from-peripheral transfers.
- Supports 1-, 2-, 4-, and 8-byte peripheral port sizes
- Supports standard 60x burst accesses (four consecutive 64-bit data phases) to/from peripherals

18.5 IDMA Transfers

The IDMA channel transfers data from a source to a destination using an intermediate transfer buffer (of programmable size) in the dual-port RAM (note that the IDMA cannot burst to or from the dual-port RAM). An efficient data-packing algorithm bursts data through the IDMA transfer buffer to minimize the bus cycles needed for the transfer; however, the IDMA will burst when it has 0x10 or more bytes to transfer. In single-address peripheral transfers, however, data is transferred directly between memory and a peripheral device without using the IDMA transfer buffer.

Unaligned data is transferred in single accesses until alignment is achieved. Then, burst transactions are used (if allowed by the user) to transfer the bulk of the data buffer. Single accesses are used again for any remaining non-burstable data at the end of the transfer.

18.5.1 Memory-to-Memory Transfers

For memory-to-memory transfers, the IDMA first fills the IDMA transfer buffer in the dual-port RAM by initiating read accesses on the source bus. It then empties the data from the internal transfer buffer to the destination bus by initiating write accesses. The transfer sizes for the source and destination buses are programmed in the IDMA parameter RAM.

For the DMA to generate bursts on the 60x bus, the address boundaries of each burst transfer must be 32-byte aligned. If the transfer does not start on a burst boundary, the IDMA controller transfers the end-of-burst (EOB) data (1–31 bytes) in non-burst transactions on the source bus and on the destination bus until reaching the next boundary. When alignment is achieved, subsequent data is bursted until the remainder of the data in the buffer is less than a burst size (32 bytes). The remaining data is transferred using non-burst transactions.

Data transfers use the parameters described in Table 18-3.

Table 18-3. IDMA Transfer Parameters

Parameter	Description
DMA_WRAP	Determines the size of the dedicated IDMA transfer buffer in dual-port RAM. The buffer size is a multiple of a 60x burst size (k×32 bytes).
SS_MAX	Initialized to (IDMA_transfer_buffer_size – 32) bytes, which is the steady-state maximum transfer size of IDMA transfer. This condition ensures that the transfer buffer is either filled by one SS_MAX bytes transfer and emptied in one or several transfers, or filled by one or several transfers to be emptied in one SS_MAX bytes transfer. In terms of bursts, if the transfer buffer contains k bursts (each is 32 bytes long), then SS_MAX equals to k – 1 bursts which is (k – 1) × 32 bytes.
STS/DTS	Source/destination transfer size. These parameters determine the access sizes in which the source/destination is accessed in steady state of work. At least one of these values (DTS/STS) must be initialized to the value of SS_MAX.

Figure 18-5 shows the IDMA transfer buffer.

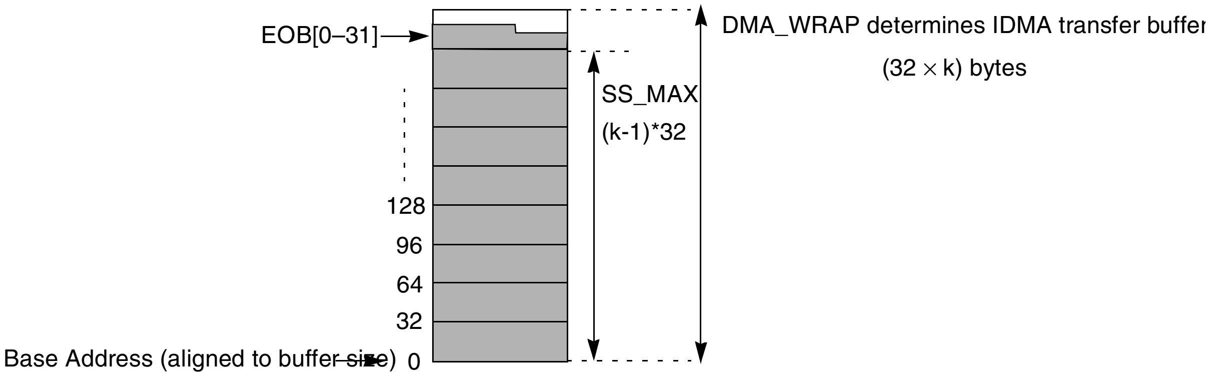

EOB[0–31]
DMA_WRAP determines IDMA transfer buffer
(32 × k) bytes
SS_MAX
(k-1)*32
128
96
64
32
Base Address (aligned to buffer size)
0

Figure 18-5. IDMA Transfer Buffer in the Dual-Port RAM

Each buffer's contents are transferred in three phases:

- First phase. The internal transfer buffer is filled with [EOB(alignment to source address) + SS_MAX] bytes, read from the source bus. Then, if EOB(alignment to destination address) ≤ EOB(alignment to source address), [EOB(destination) + SS_MAX] bytes are written from the transfer buffer to the destination bus, or if EOB(destination) > EOB(source), [EOB(destination) + (k-2)*32] bytes are written bytes are written. This write transfer size leaves a remainder of 0–31 bytes in the transfer buffer after the last write burst of the steady-state phase. After the first phase, burst alignment is ensured.
- Steady-state phase. The transfer buffer is filled with SS_MAX bytes (k – 1 bursts), read from the source bus in STS units. Then, SS_MAX bytes are written to the destination bus, in DTS units, from the transfer buffer. Because alignment is ensured from the first phase, all bus transfers are

bursts. This sequence is repeated until there are no more than SS_MAX bytes to be transferred. A remainder of 0–31 bytes is left in the transfer buffer after the last burst write.

- Last phase. The remaining data is read into the transfer buffer in bursts, with the last 1–31 bytes read in single accesses. All data in the transfer buffer is written to the destination bus in bursts, with the last 1–31 bytes written in single accesses. The last transfers, read/write or both can be accompanied with DONE assertion, if programmed.

Figure 18-6 shows an example of the three IDMA transfer stages.

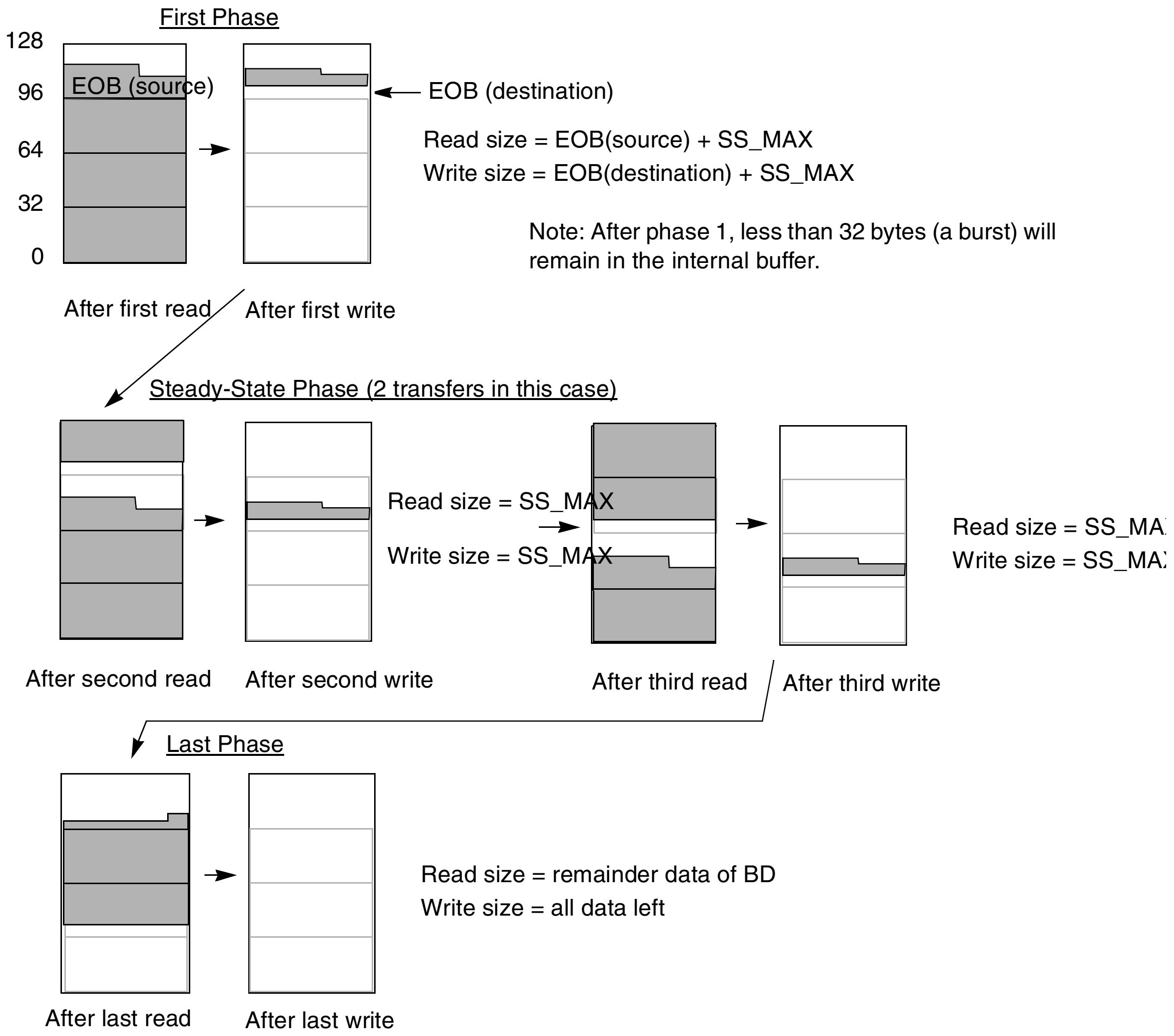


Figure 18-6. Example IDMA Transfer Buffer States for a Memory-to-Memory Transfer (Size = 128 Bytes)

18.5.1.1 External Request Mode

Memory-to-memory transfers can be configured to operate in external request mode (DCM[ERM] = 1). In external request mode, every read transfer is triggered by the assertion of DREQ. When the transfer buffer is full, the first write transfer is done automatically. Additional write transfers, if needed, are triggered by DREQ assertions. Because at least one of the transfer sizes (STS or DTS) equals SS_MAX, every DREQ assertion causes one transfer to the smaller (in STS/DTS terms) bus. If STS = DTS, asserting DREQ triggers one read transfer automatically followed by one write transfer.

NOTE

> External request mode does not support external $\overline{\text{DONE}}$ signaling from a device and $\overline{\text{DACK}}$ signaling from an IDMA channel.

18.5.1.2 Normal Mode

When external request mode is not selected (DCM[ERM] = 0), the IDMA channel operates automatically, ignoring DREQ.

18.5.1.3 Working with a PCI Bus

When working to/from the PCI bus, the data usually comes on the bus in one long burst. The alignment policy described above to support 60x bus bursts does not affect its efficiency.

18.5.2 Memory to/from Peripheral Transfers

Working with peripheral devices requires the external signals $\overline{\text{DONE}}$, DREQ, $\overline{\text{DACK}}$ to control the data transfer using the following rules:

- The peripheral sets a request for data to be read-from/write-to by asserting DREQ as configured, falling or rising edge sensitive.
- The peripheral transfers/samples the data when $\overline{\text{DACK}}$ is asserted.
- The peripheral asserts $\overline{\text{DONE}}$ to stop the current transfer.
- The peripheral terminates the current transfer when $\overline{\text{DONE}}$ is asserted, combined with $\overline{\text{DACK}}$, by the IDMA.

Peripherals are usually accessed with fixed port-size transfers. The transfer sizes (STS/DTS) related to the peripheral must be programmed to its port size; thus, every access to a peripheral yields a single bus transaction. The maximum peripheral port size is (bus_width – 8) bytes and should also evenly divide the buffer length, BD[Data Length].

A peripheral can also be configured to accept a burst per DREQ assertion. In this case, the transfer size parameter should be initialized to 32, and the accesses are made in bursts. See Table 18-8.

A peripheral can be accessed at a fixed address location or at incremental addresses. Setting DCM[SINC, DINC] in the DMA channel mode register causes the address to be incremented before the next transfer; see Section 18.8.2.1, “DMA Channel Mode (DCM).” This allows the IDMA to access a FIFO buffer the same way it does peripherals.

DCM[S/D] determines whether the peripheral is the source or destination.

Data can be transferred between a peripheral and memory in single- or dual-address accesses:

- For dual-address accesses, the data is read from the source, temporarily stored in the IDMA transfer buffer in the dual-port RAM, and then written to the destination.

- For single-address accesses (fly-by mode), the data is transferred directly between memory and the peripheral. Memory responds to the address phase, while the peripheral ignores it and responds to DACK assertions.

Any IDMA access to a peripheral uses the highest arbitration priority allowed for the DMA, providing faster bus access by bypassing other pending DMA requests.

18.5.2.1 Dual-Address Transfers

The following sections discuss various dual-address transfers.

18.5.2.1.1 Peripheral to Memory

Dual-address peripheral-to-memory data transfers are similar to memory-to-memory transfers using the three-phase algorithm; see Section 18.5.1, "Memory-to-Memory Transfers." When a peripheral asserts DREQ, data is loaded from the peripheral in port-size units to the internal transfer buffer. When the transfer buffer reaches the steady-state level, it is automatically written to the memory destination in one transfer. The source transfer size (STS) is initialized to the peripheral port size, and the destination transfer size (DTS) is initialized to SS_MAX.

External requests must be enabled (DCM[ERM] = 1) for dual-address peripheral-to-memory transfers. If DONE is asserted externally by the peripheral or if a STOP_IDMA command is issued, the current transfer stops. All data in the internal transfer buffer is written to memory in one transfer before its BD is closed, and the IDSR[EDN] or IDSR[SC] event bits are set; see Section 18.8.4, "IDMA Event Register (IDSR) and Mask Register (IDMR)."

When the peripheral controls a transfer of unknown length, initialize a large enough buffer so that the peripheral will most likely assert DONE before overflowing the buffer. When DONE is asserted, the BD is closed and interrupts are generated (if enabled). The next DREQ assertion opens the next BD if DCM[DT] is set; see Section 18.8.2.1, "DMA Channel Mode (DCM)."

18.5.2.1.2 Memory to Peripheral

Dual-address memory-to-peripheral data transfers are similar to memory-to-memory transfers using the three-phase algorithm; see Section 18.5.1, "Memory-to-Memory Transfers." STS is initialized to SS_MAX and DTS is initialized to the peripheral port size. The first DREQ peripheral assertion triggers a read of SS_MAX (or more in the first phase) bytes from the memory into the internal transfer buffer, automatically followed by a write of DTS bytes to the peripheral. Subsequent DREQ assertions trigger writes to the peripheral. When the transfer buffer has fewer than DTS bytes left, the next DREQ assertion triggers a read of SS_MAX bytes from memory, automatically followed by a write to the peripheral, and the sequence begins again.

External requests must be enabled (DCM[ERM] = 1) for dual-address peripheral-to-memory transfers. If DONE is asserted externally by the peripheral or if a STOP_IDMA command is issued, the current transfer is stopped, its BD is closed, and the IDSR[EDN] or IDSR[SC] event bits are set; see Section 18.8.4, "IDMA Event Register (IDSR) and Mask Register (IDMR)."

18.5.2.2 Single Address (Fly-By) Transfers

When DCM[FB] = 1, both peripheral-to-memory and memory-to-peripheral transfers occur in fly-by mode; see Section 18.8.2.1, “DMA Channel Mode (DCM).” In fly-by mode, an internal transfer buffer is not needed because the data is transferred directly between memory and the peripheral. Also, parameters related to the dual-port RAM bus are not relevant in fly-by mode. Each DREQ assertion triggers a transfer the size of the peripheral port. All transfers are made in single memory accesses accompanied by $\overline{\text{DACK}}$ assertion. When $\overline{\text{DONE}}$ is asserted externally or a STOP_IDMA command is issued, the current transfer is stopped, its BD is closed, and the IDSR[EDN] or IDSR[SC] event bits are set; see Section 18.8.4, “IDMA Event Register (IDSR) and Mask Register (IDMR).”

In fly-by mode, a peripheral can be configured to handle a burst per DREQ assertion if STS is programmed to 32. The first phase of the transfer aligns the data to the burst boundary so that subsequent accesses can be performed in bursts.

18.5.2.2.1 Peripheral-to-Memory Fly-By Transfers

During peripheral-to-memory fly-by transfers, the IDMA controller writes to memory while simultaneously asserting $\overline{\text{DACK}}$. The constant assertion of $\overline{\text{DACK}}$ enables the controller to write to memory as soon as the peripheral outputs data to the bus. Thus, data is transferred from a peripheral to memory in one data phase instead of two, increasing throughput.

For proper operation, STS must equal the peripheral port size.

18.5.2.2.2 Memory-to-Peripheral Fly-By Transfers

During memory-to-peripheral fly-by transfers, the IDMA controller reads from memory while simultaneously asserting $\overline{\text{DACK}}$.

The constant assertion of $\overline{\text{DACK}}$ enables the controller to read from memory as soon as the peripheral samples the data bus. Thus, data is transferred from memory to a peripheral in one data phase instead of two, increasing throughput.

For proper operation, DTS must equal the peripheral port size.

18.5.3 Controlling 60x Bus Bandwidth

STS, DTS, and SS_MAX can be used to control the 60x bus bandwidth occupied by the IDMA channel. In every mode except fly-by mode, at least one transfer size parameter (STS/DTS) must be initialized to the SS_MAX value. For memory-to-memory transfers, the other transfer size parameter can be initialized to a smaller value used to control the 60x bus bandwidth. For example, if the transfer size is N×32 bytes, each time the DMA controller wins arbitration, it transfers N bursts before releasing the bus. When SS_MAX bytes have been transferred, the controller reverts to single transactions (double-word, word, half-word, or byte).

Memory-to-memory transfer sizes must evenly divide into SS_MAX and also be a multiple of 32 (for bursting); see Table 18-7.

The size of the IDMA transfer buffer in the dual-port RAM should be determined by the largest transfer (usually SS_MAX + 32 bytes) needed by one of the buses, while the other transfer size can be programmed to control the bandwidth of the other bus.

Summarizing the above, a larger DMA transfer size provides for greater microcode efficiency and lower DMA bus latency because the DMA controller does not release the 60x bus until the transfer is completed. If the DMA priority on the 60x bus is high, however, other 60x masters may experience a high bus latency. Conversely, if the transfer size is small, the DMA requests the 60x bus more often, DMA latency increases and microcode efficiency decreases.

Example:

A channel is configured for data transfer from PCI memory to 60x memory. The PCI bus is not overloaded and can stand large bursts. Thus, the dual-port RAM buffer size is set as follows:

- 64 × 32 = 2048 bytes (DCM[DMA_WRAP] = 101), allowing maximum of 2016 (STS = 63 × 32 = 2016) bytes long bursts at the source (PCI) bus.

See Table 18-7 for valid values. For the 60x (destination) bus, two options are available:

- The 60x bus is loaded. Small bursts are preferred. Setting DTS to 1 × 32 is best for minimizing the contribution to the bus load. The dual-port RAM buffer is emptied in 63 DMA write transfers, of one burst long each, before the next long DMA read. Setting DTS to 7 × 32 is better for the channel performance, but is worse for the Bus since the dual-port RAM buffer is emptied in 9 DMA write transfers of 7 bursts each before the next long PCI DMA read.
- The 60x bus traffic is relatively low. Large bursts are preferred as long as they do not overload the bus. Setting DTS to 63 × 32 (SS_MAX) may be enough, but is too large for most of the systems because the dual-port RAM buffer would be written in one transfer to the 60x bus. On the other hand, setting DTS to 9 × 32 is the solution for a moderately loaded bus as the dual-port RAM buffer is emptied in seven DMA write transfers of nine bursts each before the next long PCI DMA read.

The IDMA transfer size parameters give high flexibility, but it is recommended to check overall system performance with different IDMA parameter settings for maximum throughput.

Note that the memory priority parameter DCM[LP] should be considered when dealing with bus bandwidth usage.

18.5.4 PCI Burst Length and Latency Control

In general, PCI burst length is larger than the 60x burst length, it is variable in length and is limited by system parameters such as latency timers. When the PCI bus is used, long bursts are preferred. The dual-port RAM buffer size, combined with the transfer size parameter (STS/DTS) of the PCI bus, are used to control its burst size. Long bursts are assigned by defining a big DPR buffer and setting the transfer size of the PCI to be equal to SS_MAX. See the example in Section 18.5.3, “Controlling 60x Bus Bandwidth.”

The PCI bus, by nature, also has a long latency that should also be considered, especially when working with peripherals, which usually require low latencies. A definition of a large dual-port RAM buffer may end in a high latency, because it takes a long time from the DREQ assertion until the buffer is filled and data is provided.

The IDMA transfer size parameters give high flexibility to the user but it recommended to check the overall performance of the system with different IDMA parameters setting for maximum throughput.

18.6 IDMA Priorities

Each IDMA channel can be programmed to have a higher or lower priority relative to the serial controllers or to have the lowest overall priority when requesting service from the CP. The IDMA priorities are programmed in RCCR[DR*x*QP]; see Section 13.3.6, “RISC Controller Configuration Register (RCCR).” Take care to avoid overrun or underrun errors in the serial controllers when selecting high priorities for IDMA.

Additional priority over all serial controllers can be selected by setting DCM[LP]; see Section 18.8.2.1, “DMA Channel Mode (DCM).”

18.7 IDMA Interface Signals

Each IDMA has three dedicated handshake control signals for transfers involving an external peripheral device: DMA request (DREQ[2–3]), DMA acknowledge ($\overline{\text{DACK[2–3]}}$) and DMA done ($\overline{\text{DONE[2–3]}}$). DREQ*x* may also be used to control the transfer pace of memory-to-memory transfers.

- DREQ*x* is the external DMA request signal.
- $\overline{\text{DACK}x}$ is the DMA acknowledge.
- $\overline{\text{DONE}x}$ marks the end of an IDMA transfer.

The IDMA signals are multiplexed with other internal controller signals at the parallel I/O ports. To enable the IDMA signals, the corresponding bits in the parallel I/O registers should be set. See Chapter 37, “Parallel I/O Ports.”

18.7.1 DREQ*x* and $\overline{\text{DACK}x}$

When the peripheral requires IDMA service, it asserts DREQ*x* and the PowerQUICC II begins the IDMA process. When the IDMA service is in progress, $\overline{\text{DACK}x}$ is asserted during accesses to the peripheral. A peripheral must validate the transfer by asserting $\overline{\text{TA}}$ or signal an error by asserting $\overline{\text{TEA}}$.

If the user programs the memory controller for the peripheral, the PowerQUICC II asserts $\overline{\text{TA}}$ so that the peripheral terminates $\overline{\text{DACK}x}$. Without $\overline{\text{TA}}$ assertion, $\overline{\text{DACK}x}$ could be asserted for only one cycle, and no data transfer occurs. To avoid peripherals mistaking this as a valid data transfer, $\overline{\text{DACK}x}$ should be qualified with $\overline{\text{TA}}$.

NOTE

Programming the parallel ports DREQ pins generates a transition on the internal DREQ signals. This may cause an IDMA transaction, and, if the IDMA is not initialized at that time, the IDMA transaction may lock the CPM. Therefore, do one of the following:

- Program the parallel ports to be DREQ after initializing the IDMA registers and parameter RAM.

- Pull down (pulling up does not help) the DREQ inputs before programming the parallel port DREQ pins and until after setting the IDMA registers, or program the IDMA registers for a dummy transaction before programming the parallel port DREQ pins.

DREQ*x* may be configured as either edge- or level-sensitive by programming the RCCR[DR*x*M]. When DREQ*x* is configured as edge-sensitive, RCCR[EDM*x*] controls whether the request is generated on the rising or falling edge; see Section 13.3.6, “RISC Controller Configuration Register (RCCR).”

DREQ*x* is sampled at each rising edge of the clock to determine when a valid request is asserted by the device.

18.7.1.1 Level-Sensitive Mode

For external devices requiring very high data transfer rates, level-sensitive mode allows the IDMA to use a maximum bandwidth to service the device. The device requests service by asserting DREQ*x* and leaving it asserted as long as it needs service. This mode is selected by setting the corresponding RCCR[DR*x*M].

The IDMA asserts $\overline{\text{DACK}}$ each time it issues a bus transaction to either read or write the peripheral. The peripheral must use $\overline{\text{TA}}$ and $\overline{\text{TEA}}$ for data validation. $\overline{\text{DACK}}$ is the acknowledgment of the original burst request given on DREQ*x*. DREQ*x* should be negated during the $\overline{\text{DACK}}$ active period to ensure that no further transactions are performed.

18.7.1.2 Edge-Sensitive Mode

For external devices that generate a pulsed signal for each operand to be transferred, edge-sensitive mode should be used. In edge-sensitive mode, the IDMA controller moves one operand for each falling/rising (as configured by RCCR[EDM*x*]) edge of DREQ*x*. This mode is selected by clearing the corresponding RCCR[DR*x*M] and programming the corresponding RCCR[EDM*x*] to the proper edge.

When the IDMA controller detects a valid edge on DREQ*x*, a request becomes pending and remains pending until it is serviced by the IDMA. Subsequent changes on DREQ*x* are ignored until the request begins to be serviced. The servicing of the request results in one operand being transferred. Each time the IDMA issues a bus transaction to either read or write the device, the IDMA asserts $\overline{\text{DACK}}$. The device must use $\overline{\text{TA}}$ and $\overline{\text{TEA}}$ for data validation. Thus, $\overline{\text{DACK}}$ is the acknowledgment of the original transaction request given on DREQ*x*.

18.7.2 $\overline{\text{DONE}x}$

This bidirectional open-drain signal is used to indicate the last IDMA transfer. $\overline{\text{DONE}}$ can be an output of the IDMA in the source or destination bus transaction if the transfer count is exhausted. This function is controlled by BD[SDN, DDN].

$\overline{\text{DONE}}$ can also operate as an input. When operating in external request modes, $\overline{\text{DONE}}$ may be used as an input to the IDMA controller to indicate that the device being serviced requires no more transfers. In that case, the transfer is terminated, the current BD is closed, and an interrupt is generated (if enabled).

NOTE

> $\overline{\text{DONE}}$ is ignored if it is asserted externally during internal request mode (DCM[ERM] = 0).
>
> $\overline{\text{DONE}}$ must not be asserted externally during memory-to-memory transfers if external request mode is enabled (DCM[ERM] = 1).

18.8 IDMA Operation

Every IDMA operation involves the following steps—IDMA channel initialization, data transfer, and block termination.

- During initialization, the core initializes the IDMA_BASE register in the internal parameter RAM to point to the IDMA-specific table in RAM. This table contains control information for the IDMA operation. In addition the core initializes the parallel I/O registers to enable IDMA external signals, if needed, and other registers related to the channel priority and operation modes; see Section 18.11, “Programming the Parallel I/O Registers.” The core initiates the IDMA BDs to point to the data for the transfer and/or a free space for data to be transferred to, and starts the transfer by issuing the START_IDMA command.
- During data transfer, the IDMA accepts requests for data transfers and provides addressing and bus control for the transfers.
- Termination occurs when the IDMA operation completes or the peripheral asserts $\overline{\text{DONE}}$ externally. The core can initiate termination by using the STOP_IDMA command. The IDMA can interrupt the core if interrupts are enabled to signal for operation termination and other events related to the data transfer.

The IDMA uses a data structure, which, as with serial controller BDs, allows flexible data allocation and eliminates the need for core intervention between transfers. BDs contain information describing the data block and special control options for the DMA operation while transferring the data block.

18.8.1 Auto Buffer and Buffer Chaining

The core processor should initialize the IDMA BD table with the appropriate buffer handling mode, source address, destination address, and block length. See Figure 18-7.

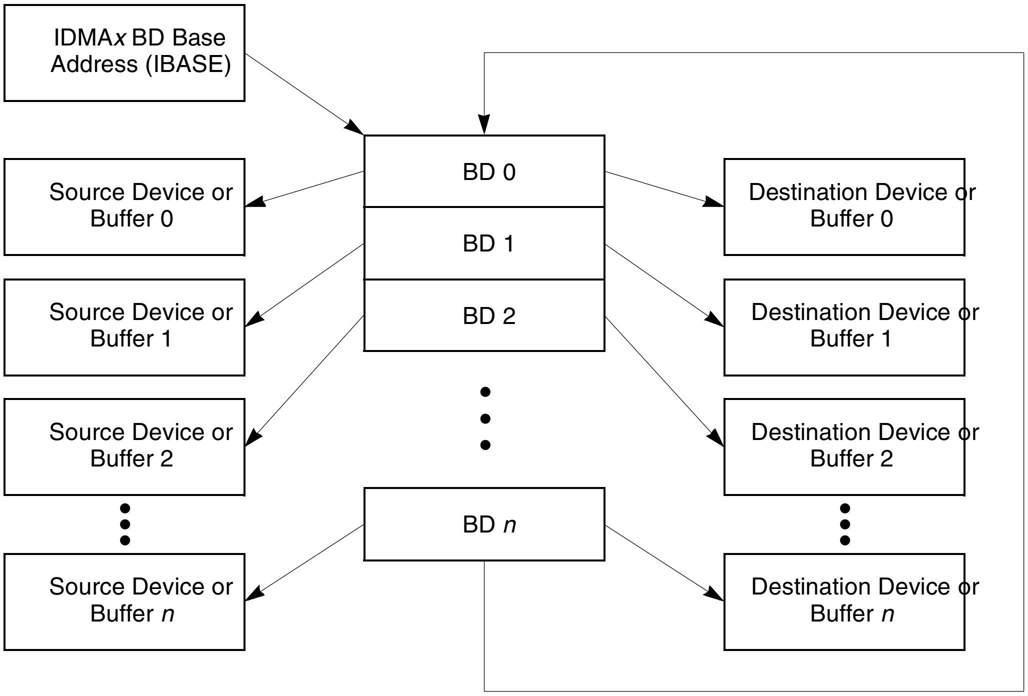

IDMA*x* BD Base Address (IBASE)
BD 0
BD 1
BD 2
BD *n*
Source Device or Buffer 0
Source Device or Buffer 1
Source Device or Buffer 2
Source Device or Buffer *n*
Destination Device or Buffer 0
Destination Device or Buffer 1
Destination Device or Buffer 2
Destination Device or Buffer *n*

Figure 18-7. IDMA*x* Channel's BD Table

Data associated with each IDMA channel is stored in buffers and each buffer is referenced by a BD that uses a circular table structure in the dual-port RAM. Control options such as interrupt and $\overline{\text{DONE}}$ assertion are also programmed on a per-buffer basis in each BD.

Data may be transferred in the two following modes:

- Auto buffer mode—The IDMA continuously transfers data to/from the location programmed in the BD until a STOP_IDMA command is issued or $\overline{\text{DONE}}$ is asserted externally.
- Buffer chaining mode—Data is transferred according to the first BD parameters, then the second BD and so forth. The first BD is reused (if ready) until the BD with the last bit set is reached. IDMA transfers stop and restarts when the BD table is reinitialized and a START_IDMA command is issued.

18.8.2 IDMA*x* Parameter RAM

When an IDMA*x* channel is configured to auto buffer or buffer chaining mode, the PowerQUICC II uses the IDMA*x* parameters listed in the Table 18-4. Parameters should be modified only while the channel is disabled, that is, before the first START_IDMA command or when the event register's stop-completed bit (IDSR[SC]) is set following a STOP_IDMA command.

Each IDMA*x* channel parameter table can be placed at any 64-byte aligned address in the dual-port RAM's general-purpose area (banks 1–8, 11 and 12). The CP accesses each IDMA*x* channel parameter table using a user-programmed pointer (IDMA*x*_BASE) located in the parameter RAM; see Section 13.5.2, "Parameter RAM." For example, if the IDMA2 channel parameter table is to be placed at address offset 0x2000 in the dual-port RAM, write 0x2000 to IDMA2_BASE.

Table 18-4. IDMA*x* Parameter RAM

Offset [1]	Name	Width	Description
0x00	IBASE	Hword	IDMA BD table base address. Defines the starting location in the dual-port RAM for the set of IDMA BDs. It is an offset from the beginning of the dual-port RAM. The user must initialize IBASE before enabling the IDMA channel and should not overlap BD tables of two enabled serial controllers or IDMA channels or erratic operation results. IBASE should be 16-byte aligned.
0x02	DCM	Hword	DMA channel mode. See Section 18.8.2.1, "DMA Channel Mode (DCM)."
0x04	IBDPTR	Hword	IDMA BD pointer. Points to the current BD during transfer processing. Points to the next BD to be processed when an idle channel is restarted. Initialize to IBASE before the first START_IDMA command. If BD[W] = 1, the CP initializes IBPTR to IBASE When the end of an IDMA BD table is reached. After a STOP_IDMA command is issued, IBDPTR points to the next BD to be processed. It can be modified after SC interrupt is set and before a START_IDMA command is reissued.
0x06	DPR_BUF	Hword	IDMA transfer buffer base address. The base address should be aligned according to the buffer size determined by DCM[DMA_WRAP]. The transfer buffer size should be consistent with DCM[DMA_WRAP]; that is, DPR_BUF = (64 X $2^{\text{DMA\_WRAP}}$). See Section 18.8.2.1, "DMA Channel Mode (DCM)."
0x08	BUF_INV	Hword	Internal buffer inventory. Indicates the quantity of data inside the internal buffer.
0x0A	SS_MAX	Hword	Steady-state maximum transfer size in bytes. User-defined parameter to increase microcode efficiency. Initialize to internal_buffer_size – 32, that is, SS_MAX = (64 × $2^{\text{DMA\_WRAP}}$) - 32. If possible, SS_MAX is used as the transfer size on transfers to/from memory in memory-to-peripheral mode or in peripheral-to-memory mode. For memory-to-memory mode, SS_MAX is used as the transfer size for at least one of the devices. SS_MAX should be consistent with STS, DTS, and DCM[S/D]. See Table 18-7 and Table 18-8.
0x0C	DPR_IN_PTR	Hword	Write pointer inside the internal buffer.
0x0E	STS	Hword	Source transfer size in bytes. All transfers from the source (except the start alignment and the end) are written to the bus using this parameter. In memory-to-peripheral mode, STS should be initialized to SS_MAX. In peripheral-to-memory mode, STS should be initialized to the peripheral port size or peripheral transfer size (if the peripheral accepts bursts). See Table 18-8 for valid STS values for peripherals. In fly-by mode, STS is initialized to the peripheral port size. In memory-to-memory mode: • STS should be initialized to SS_MAX. • DTS value should be initialized to SS_MAX. STS can be initialized to values other than SS_MAX in the following conditions: – STS must divide SS_MAX. – STS must be divided by 32 to enable bursts during the steady-state phase. • See Table 18-7 for memory-to-memory valid STS values.
0x10	DPR_OUT_PTR	Hword	Read pointer inside the internal buffer.
0x12	SEOB	Hword	Source end of burst. Used for alignment of the first read burst.
0x14	DEOB	Hword	Destination end of burst. Used for alignment of the first write burst.

Table 18-4. IDMA*x* Parameter RAM (continued)

Offset [1]	Name	Width	Description
0x16	DTS	Hword	Destination transfer size in bytes. All transfers to destination (except the start alignment and the tail) are written to the bus using this parameter. In peripheral-to-memory mode, DTS should equal SS_MAX. In memory-to-peripheral modes, initialize DTS to the peripheral port size if transfer's destination is a peripheral. Valid sizes for peripheral destination is 1, 2, 4, and 8 bytes, or peripheral transfer size (if the peripheral accepts bursts). See Table 18-8 for valid STS values for peripherals. In fly-by mode, DTS is initialized to the peripheral port size. • In memory-to-memory mode: • DTS value is initialized to SS_MAX. • STS value is initialized to SS_MAX. DTS can be initialized to values other than SS_MAX in the following conditions: – DTS must divide SS_MAX. – DTS must be divided by 32, to enable bursts in steady-state phase. • See Table 18-8 for valid memory-to-memory DTS values.
0x18	RET_ADD	Hword	Used to save return address when working in ERM = 1 mode
0x1A	—	Hword	Reserved, should be cleared
0x1C	BD_CNT	Word	Internal byte count
0x20	S_PTR	Word	Source internal data pointer
0x24	D_PTR	Word	Destination internal data pointer
0x28	ISTATE	Word	Internal. Should be cleared before every START_IDMA command.

[1] From the pointer value programmed in IDMA*x*_BASE: IDMA2_BASE at 0x88FE and IDMA3_BASE at 0x89FE; see Section 13.5.2, "Parameter RAM."

18.8.2.1 DMA Channel Mode (DCM)

The IDMA channel mode (DCM), shown in Figure 18-8, is a 16-bit field within the IDMA parameter RAM, that controls the operation modes of the IDMA channel. As are all other IDMA parameters, the DCM is undefined at reset.

	0	1	2 4	5	6	7 9	10	11	12	13	14 15
Field	FB	LP	—	TC2	—	DMA_WRAP	SINC	DINC	ERM	DT	S/D
Reset	—										
R/W	R/W										

Figure 18-8. DCM Parameters

Table 18-5 describes DCM bits.

Table 18-5. DCM Field Descriptions

Bits	Name	Description
0	FB	Fly-by mode. See Table 18-6. 0 Dual-address mode. 1 Fly-by (single-address) mode. The internal IDMA transfer buffer is not used. Valid only in peripheral-to-memory (S/D=10) or memory-to-peripheral (S/D=01) modes.
1	LP	Low priority. Applies to memory-to-memory accesses only. See Section 4.3.2, “System Configuration and Protection Registers.” 0 The IDMA transaction to memory is in middle CPM request priority. 1 The IDMA transaction to memory is in low CPM request priority. Note that IDMA single-address (fly-by) transfers with external peripherals are always high priority, ignoring this bit and bypassing other pending SDMA requests.
2–4	—	Reserved, should be cleared.
5	TC2	Driven on TC[2] during IDMA transactions. The TC[0:1] signals are always driven to 0b11 during IDMA transactions.
6	—	Reserved, should be cleared.
7–9	DMA_WRAP	DMA wrap. Defines the size of the IDMA transfer buffer. The IDMA pointer wraps to the beginning of the buffer whenever DMA_WRAP bytes have been transferred to/from the buffer. 000 64 byte 001 128 byte 010 256 byte 011 512 byte 100 1024 byte 101 2048 byte 11x Reserved Table 18-7 and Table 18-8 describes the relations between the parameter’s initial value and SS_MAX, STS, DTD and DCM[S/D] parameters. The IDMA transfer buffer (DPR_BUF) size should be consistent with DCM[DMA_WRAP]; that is DPR_BUF = 64 $\times$ $2^{(DMA\_WRAP)}$.
10	SINC	Source increment address. 0 Source address pointer (S_PTR) is not incremented in the source read transaction. Should be cleared for peripheral-to-memory transfers if the peripheral has a fixed address. 1 CP increments the source address pointer (S_PTR) with the number of bytes transferred in the source read transaction. Used for memory-to-memory and memory-to-peripheral transfers. In fly-by mode, SINC controls the memory address increment and should equal DINC.
11	DINC	Destination increment address. 0 Destination address pointer (D_PTR) is not changed in the destination write transaction. Used for memory-to-peripheral transfers if the peripheral has a fixed address. 1 CP increments the destination pointer (D_PTR) with the number of bytes transferred in the destination write transaction. Used for memory-to-memory and memory-to-peripheral transfers. In fly-by mode, DINC should equal SINC.

Table 18-5. DCM Field Descriptions (continued)

Bits	Name	Description
12	ERM	External request mode. 0 The CP transfers continuously, as if an external level request is asserted, regardless of the DREQ signal assertion. The CP stops the transfer when there are no more valid BDs or after a STOP_IDMA command is issued. DONE assertion by a external device is ignored. 1 The CP responds to DREQ as configured (edge/level) by performing single- or dual-address transfers. The CP also responds to DONE assertions. **Note:** Memory-to-memory transfers (S/D=00) with external request (ERM=1) is allowed, but DONE assertion is not supported in this mode (DONE should be disabled).
13	DT	DONE treatment: 0 After external DONE assertion, the IDMA ignores further DREQ assertions. The CP closes the current BD and IDMA stops. START_IDMA command should be issued before assertion of another DREQ. 1 After external DONE assertion, the CP closes the current BD. The IDMA continues to the next BD when DREQ is asserted.
14–15	S/D	Source/destination is a peripheral device or memory. See Table 18-6. 00 Read from memory, write to memory. 10 Read from peripheral, write to memory. 01 Read from memory, write to peripheral. 11 Reserved When a device is a peripheral: • DACK is asserted during transfers to/from it. • It may assert DONE to terminate all accesses to/from it. • It can be operated in fly-by mode—respond to DACK ignoring the address. • It gets highest DMA priority on the bus arbiter and the lowest DMA latency available.

18.8.2.2 Data Transfer Types as Programmed in DCM

Table 18-6 summarizes the types of data transfers according to the DCM programming.

Table 18-6. IDMA Channel Data Transfer Operation

S/D	FB	Read From	Write To	Description (Steady-State Operation)
01	0	Memory (STS = SS_MAX)	Peripheral (DTS = port size or 32)	Read from memory: Filling internal buffer in one DMA transfer. On the bus: one burst or more, depends on STS
				Write to peripheral: In smaller transfers until internal buffer empties. On the bus: singles or burst, depends on DTS
10	0	Peripheral (STS = port size or 32)	Memory (DTS = SS_MAX)	Read from peripheral: Filling internal buffer in several DMA transfers. On the bus: singles or burst, depends on STS
				Write to memory: in one DMA transfer, internal buffer empties. On the bus: one burst or more, depends on DTS
00	0	Memory (STS = SS_MAX)	Memory (DTS = SS_MAX or less)	Read from memory: Filling internal buffer in one DMA transfer. On the bus: one burst or more, depends on STS
				Write to memory: in one transfer or more until internal buffer empties. On the bus: singles or bursts, depends on DTS

Table 18-6. IDMA Channel Data Transfer Operation (continued)

S/D	FB	Read From	Write To	Description (Steady-State Operation)
00	0	Memory (STS = SS_MAX or less)	Memory (DTS = SS_MAX)	Read from memory: Filling internal buffer in one or more DMA transfers. On the bus: singles or bursts, depends on STS
				Write to memory: in one DMA transfer, internal buffer empties. On the bus: one burst or more, depends on DTS
01	1	Memory to peripheral (DTS = port size or 32)	—	Read transaction from memory while asserting $\overline{\text{DACK}}$ to peripheral. Peripheral samples the data read from memory. On the bus: singles or bursts, depends on DTS
10	1	—	Peripheral to memory (STS = port size or 32)	Write transaction to memory while asserting $\overline{\text{DACK}}$ to peripheral. Peripheral provides the data that is written to the memory. On the bus: singles or bursts, depends on STS

18.8.2.3 Programming DTS and STS

The options for setting STS and DTS depend on (DCM[DMA_WRAP]) and are described in the following tables for memory/memory and memory/peripheral transfers.

Table 18-7 describes valid STS/DTS values for memory-to-memory operations.

Table 18-7. Valid Memory-to-Memory STS/DTS Values

DMA_WRAP	Internal Buffer Size	SS_MAX	STS (in Bytes)	DTS (in Bytes)	Number of Transfers to Fill Internal Buffer	
					STS Size	DTS Size
000	64	1 × 32	1 × 32	32	1	1
			32	1 × 32	1	1
001	128	3 × 32	3 × 32	3 × 32, 32	1	1, 3
			3 × 32, 32	3 × 32	1, 3	1
010	256	7 × 32	7 × 32	7 × 32, 32	1	1, 7
			7 × 32, 32	7 × 32	1, 7	1
011	512	15 × 32	15 × 32	15 × 32, 3 × 32, 5 × 32, 32	1	1, 5, 3, 15
			15 × 32, 3 × 32, 5 × 32, 32	15 × 32	1, 5, 3, 15	1
100	1024	31 × 32	31 × 32	31 × 32, 32	1	1, 31
			31 × 32, 32	31 × 32	1, 31	1

Table 18-7. Valid Memory-to-Memory STS/DTS Values (continued)

DMA_WRAP	Internal Buffer Size	SS_MAX	STS (in Bytes)	DTS (in Bytes)	Number of Transfers to Fill Internal Buffer	
					STS Size	DTS Size
101	2048	63 × 32	63 × 32	63 × 32, 9 × 32, 7 × 32, 32	1	1, 7, 9, 63
			63 × 32, 9 × 32, 7 × 32, 32	63 × 32	1, 7, 9, 63	1

Table 18-8 describes valid STS/DTS values for memory/peripheral operations.

Table 18-8. Valid STS/DTS Values for Peripherals

DMA_WRAP	Internal Buffer Size	SS_MAX	S/D Mode	STS (in Bytes)	DTS (in Bytes)
000	64	1 × 32	01	1 × 32	1, 2, 4, 8 (single)[1]; 32 (burst)[2]
			10	1, 2, 4, 8 (single); 32 (burst)	1 * 32
001	128	3 × 32	01	3 × 32	1, 2, 4, 8 (single); 32 (burst)
			10	1, 2, 4, 8 (single); 32 (burst)	3 * 32
010	256	7 × 32	01	7 × 32	1, 2, 4, 8 (single); 32 (burst)
			10	1, 2, 4, 8 (single); 32 (burst)	7 * 32
011	512	15 × 32	01	15 × 32	1, 2, 4, 8 (single); 32 (burst)
			10	1, 2, 4, 8 (single); 32 (burst)	15 * 32
100	1024	31 × 32	01	31 × 32	1, 2, 4, 8 (single); 32 (burst)
			10	1, 2, 4, 8 (single); 32 (burst)	31 * 32
101	2048	63 × 32	01	63 × 32	1, 2, 4, 8 (single); 32 (burst)
			10	1, 2, 4, 8 (single); 32 (burst)	63 * 32

[1] These values come out as a single transaction on the bus.

[2] Peripherals that can accept bursts of 32 bytes are supported.

18.8.3 IDMA Performance

The transfer parameters STS, DTS, SS_MAX, and DMA_WRAP determine the amount of data transferred for each START_IDMA command issued. Using large internal IDMA transfer buffers and the maximum transfer sizes allows longer transfers to memory devices, optimizes bus usage and thus reduces the overall load on the CP.

For example, 2,016 bytes can be transferred by issuing one START_IDMA command using a 2-Kbyte internal transfer buffer, or by issuing 63 START_IDMA commands using a 64-byte buffer. The load on the CP in the second case is about 63 times more than the first.

18.8.4 IDMA Event Register (IDSR) and Mask Register (IDMR)

The IDMA event (status) register (IDSR) is used to report events recognized by the IDMA controller. On recognition of an event, the controller sets the corresponding IDSR bit. Each IDMA event bit can generate a maskable interrupt to the core. Even bits are cleared by writing ones; writing zeros has no effect.

The IDMA mask register (IDMR) has the same format as IDSR. Setting IDMR bits enables, and clearing IDMR bits disables, the corresponding interrupts in the event register.

Figure 18-9 shows the bit format for IDSR and IDMR.

<table>
<tr><td></td><td>0 3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>Field</td><td>—</td><td>SC</td><td>OB</td><td>EDN</td><td>BC</td></tr>
<tr><td>Reset</td><td colspan="5">0000_0000</td></tr>
<tr><td>R/W</td><td>R</td><td colspan="4">R/W</td></tr>
<tr><td>Addr</td><td colspan="5">0x11028 (IDSR2), 0x11030 (IDSR3)/
0x1102C (IDMR2), 0x11034 (IDMR3)</td></tr>
</table>

Figure 18-9. IDMA Event/Mask Registers (IDSR/IDMR)

Table 18-9 describes IDSR/IDMR fields.

Table 18-9. IDSR/IDMR Field Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	SC	Stop completed. Set after the IDMA channel completes processing the STOP_IDMA command. Do not change channel parameters until SC is set.
5	OB	Out of buffers. Set to indicate that the IDMA channel encountered no valid BDs for the transfer.
6	EDN	External $\overline{\text{DONE}}$ was asserted by device. Set to indicate that the IDMA channel terminated a transfer because $\overline{\text{DONE}}$ was asserted by an external device, on the former SDMA transaction.
7	BC	BD completed. Set only after all data of a BD whose I (interrupt) bit is set has completed transfer to the destination.

18.8.5 IDMA BDs

Source addresses, destination addresses, and byte counts are presented to the CP using the special IDMA BDs. The CP reads the BDs, programs the SDMA channel, and notifies the core about the completion of a buffer transfer using the IDMA BDs. This concept is similar to the one used for the serial controllers on the PowerQUICC II except that the BD is larger because it contains additional information.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**V**	—	**W**	**I**	**L**	—	**CM**	—		**SDN**	**DDN**	**DGBL**	**DBO**		—	**DDTB**
Offset + 2	—		**SGBL**	**SBO**		—	**SDTB**	—								
Offset + 4	**Data Length**															
Offset + 6																
Offset + 8	**Source Data Buffer Pointer**															
Offset + A																
Offset + C	**Destination Data Buffer Pointer**															
Offset + E																

Figure 18-10. IDMA BD Structure

Table 18-10 describes IDMA BD fields.

Table 18-10. IDMA BD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid 0 This BD does not contain valid data for transfer. 1 This BD contain valid data for transfer. The CP checks this bit before starting a BD service. If this bit is cleared when the CP accesses the BD, an interrupt IDSR[OB] is issued to the core, the IDMA channel is stopped until a START_IDMA command is issued. After the BD is serviced this bit is cleared by CP unless CM = 1.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the BD table. 1 Last BD in the table. After the associated buffer has been used, the CP transfers data from the first BD in the table, which is pointed by IBASE. The number of BDs in this table is programmable and determined by W bit and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 When the CP services all the buffer’s data, IDSR[BC] is set, which generates a maskable interrupt.
	4	**L**	Last 0 Not the last buffer of a chain to be transferred in buffer chaining mode. The I bit can be used to generate an interrupt when this buffer service is complete. 1 Last buffer of a chain to be transferred in buffer chaining mode. When this BD service is complete the channel is stopped by CP until START_IDMA command is issued. This bit should be set only in buffer chaining mode (CM bit 6 = 0).

Table 18-10. IDMA BD Field Descriptions (continued)

Offset	Bits	Name	Description
	5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Buffer chaining mode. The CP clears V after this BD is serviced. Buffer chaining mode is used to transfer large quantities of data into non-contiguous buffer areas. The user can initialize BDs ahead of time, if needed. The CP automatically loads the IDMA registers from the next BD values when the transfer is terminated. 1 Auto buffer mode (continuous mode). The CP does not clear V after this BD is serviced. This is the only difference between auto buffer mode and buffer chaining mode. Auto buffer mode transfers multiple groups of data to/from a buffer table and does not require BD reprogramming. The CP automatically reloads the IDMA registers from the next BD values when the transfer is terminated. Either a single BD or multiple BDs can be used to create an infinite loop of repeated data moves. **Note:** The I bit can still be used to generate an interrupt in this mode.
	7–8	—	Reserved, should be cleared.
	9	**SDN**	Source done 0 $\overline{\text{DONE}}$ is inactive during this BD. 1 The IDMA asserts $\overline{\text{DONE}}$ at the last read data phase of the BD. In fly-by mode (DCM[FB] = 1), SDN should be same as DDN.
	10	**DDN**	Destination done 0 $\overline{\text{DONE}}$ is inactive during this BD. 1 The IDMA asserts $\overline{\text{DONE}}$ at the last write data phase of the BD. In fly-by mode (DCM[FB] = 1), DDN should be same as SDN.
	11	**DGBL**	Destination global 0 Snooping is not activated. 1 Snooping is activated for write transactions to the destination. In fly-by mode, should be the same as SGBL
	12–13	**DBO**	Destination byte ordering: 01 Munged little endian 1x Big endian (Freescale) 00 Reserved In fly-by mode, should be the same as SBO
	14	—	Reserved, should be cleared
	15	**DDTB**	Destination data bus. 0 The destination address lies within the 60x bus. 1 Reserved In fly-by mode, should be the same as SDTB
0x02	0–1	—	Reserved, should be cleared
	2	**SGBL**	Source global 0 Snooping is not activated. 1 Snooping is activated for read transactions from the source. In fly-by mode, should be the same as DGBL.
	3–4	**SBO**	Source byte ordering: 01 Munged little endian 1x Big endian (Freescale) 00 Reserved In fly-by mode, should be the same as DBO

Table 18-10. IDMA BD Field Descriptions (continued)

Offset	Bits	Name	Description
	5	—	Reserved, should be cleared
	6	**SDTB**	Source data bus. 0 The source address lies within the 60x bus. 1 The source address lies within the PCI bus. In fly-by mode, should be the same as DDTB
	7–15	—	Reserved, should be cleared
0x04	0–31	**Data Length**	Number of bytes the IDMA transfers. Should be programmed to a value greater than zero. **Note:** When operating with a peripheral that accepts only single bus transactions (transfer size < 32), data length should be a multiple of the peripheral transfer size (STS for S/D = 10, or DTS for S/D = 01). Also, there is no error notification if the data length does not match the buffer sizes.
0x08	0–31	**Source Buffer Pointer**	Holds the address of the associated buffer. Buffers may reside in internal or external memory. Note that if the source/destination is a device, the pointer should contain the device address. In fly-by mode, the pointers should contain the memory address.
0x0C	0–31	**Destination Buffer Pointer**	

18.9 IDMA Commands

The user has two commands to control each IDMA channel. These commands are executed through the CP command register (CPCR); see Section 13.4, “Command Set.”

18.9.1 START_IDMA Command

The START_IDMA command is used to start a transfer on an IDMA channel. The user must initialize all parameters relevant for the correct operation of the channel (IDMAx_BASE and IDMA channel parameter table) before issuing this command.

To restart the channel operation, the START_IDMA command can be reissued after every pause in channel activity. The user must ensure that parameters are correct for the channel to continue operation correctly.

The parameter ISTATE of the IDMA parameter RAM should be cleared before every issue of a START_IDMA command.

An IDMA pause may occur for one of the following reasons:

- The channel is out of buffers—IDSR[OB] event is set and an interrupt is generated to the core, if enabled.
- $\overline{\text{DONE}}$ was asserted externally and DCM[DT] = 0 (see Table 18-5). An IDSR[EDN] event is set and an interrupt is generated to the core, if enabled.
- The STOP_IDMA command was issued.
- The channel has finished a transfer of a BD with the last bit (L) set.

If the START_IDMA command is reissued and channel has more buffers to transfer, it restarts transferring data according to the next BD in the buffer table.

In external request mode (ERM=1), the START_IDMA command initializes the channel, but the first data transfer is performed after external DREQ*x* assertion.

In internal request mode (ERM=0), the START_IDMA command starts the data transfer almost immediately, with a delay that depends on the CP load.

18.9.2 STOP_IDMA Command

The STOP_IDMA command is issued to stop the transfer of an IDMA channel.

When a STOP_IDMA command is issued, the CP terminates current IDMA transfers and the current BD is closed (if it was open). If memory is the destination, all data in the IDMA internal buffer is transferred to memory before termination.

At the end of the stop process, the stop-completed event (SC) is set and a maskable interrupt is generated to the core. The user should not modify channel parameters until SC = 1. When the channel is stopped, it does not respond to external requests. If a START_IDMA command is reissued, the next BD in the BD table is processed (if it is valid).

In external request mode (ERM = 1), STOP_IDMA command processing has priority over a peripheral asserting $\overline{\text{DONE}}$.

Note: In memory-to-peripheral, peripheral-to-memory, and fly-by modes, if a STOP_IDMA command is issued with no data in the internal buffer, the BD is immediately closed and the channel is stopped. In this case, a peripheral expecting $\overline{\text{DONE}}$ to be asserted is not notified because the last transfer of the buffer (with BD[DDN or SDN] set) is not performed.

18.10 IDMA Bus Exceptions

Bus exceptions can occur while the IDMA has the bus and is transferring operands. In any computer system, a hardware failure can cause an error during a bus transaction due to random noise or an illegal access. When a synchronous bus structure (like those supported by the PowerQUICC II) is used, it is easy to make provisions for a bus master to detect and respond to errors during a bus transaction. The IDMA recognizes the same bus exceptions as the core, reset and transfer error, as described in Table 18-11.

Table 18-11. IDMA Bus Exceptions

Exception	Description
Reset	On an external reset, the IDMA immediately aborts the channel operation, returns to the idle state, and clears IDSR. If reset is detected when a bus transaction is in progress, the transaction is terminated, the control and address/data pins are three-stated, and bus mastership is released.
Transfer error	When a fatal error occurs during a bus transaction, a bus error exception is used to abort the transaction and systematically terminate channel operation. The IDMA terminates the current bus transaction, signals an error in the SDSR, and signals an interrupt if the corresponding bit in the SDMR is set. The CPM must be reset before IDMA operation is restarted. Any data previously read from the source into the internal storage is lost, however, issuing a START_IDMA command transfers the last BD again. Note: Any source or destination device for an operand under IDMA handshake control for single-address transfers may need to monitor $\overline{\text{TEA}}$ to detect a bus exception for the current bus transaction. $\overline{\text{TEA}}$ terminates the transaction immediately and negates $\overline{\text{DACK}}$, which is used to control the transfer to/from the device.

18.10.1 Externally Recognizing IDMA Operand Transfers

The following ways can be used determine externally that the IDMA is executing a bus transaction:

- The TC[2] signal (programmed in DCM[TC2]) or SDMA channels can be programmed to a unique code that identifies an IDMA transfer.
- The $\overline{\text{DACK}}$ signal shows accesses to the peripheral device. $\overline{\text{DACK}}$ activates on either the source or destination bus transactions, depending on DCM[S/D].

18.11 Programming the Parallel I/O Registers

The parallel I/O registers control the use of the external pins of the chip. Each pin can be used for different purposes. See Table 18-12 and Table 18-13 for the proper parallel I/O register programming dedicating the proper external ports to the four IDMA channels' external I/O signals.

Each port is controlled by five I/O registers: PPAR, PSOR, PDIR, PODR, and PDAT. Each bit in these registers controls the external pin of the same location.

- PPARC selects the pins general purpose(0)/dedicated(1) mode for port C.
- PDIRC select the pins input or inout (0)/output(1) mode for port C.
- PODRC selects the open drain pins for port C.
- PSORC selects the pins dedicated1(0)/dedicated2(1) mode for port C.
- PPARA, PDIRA, PODRA, and PSORA control port A in the same way.
- PPARD, PDIRD, PODRD, and PSORD control port D in the same way.

- The default is the value that is seen by the IDMA channel on the pin (input or inout mode only—PDIR[PN] = 0) if a PSOR*x* register bit is set to the complement value of the value in Table 18-12 and Table 18-13. See Section 37.2, "Port Registers."

Table 18-12. Parallel I/O Register Programming—Port C

Channel	Signal	Pin	PPARC	PDIRC	PODRC	PSORC	Default
IDMA2	DREQ2 (I)	PC[24]	1	0	0	1	GND
	DACK2 (O)	PC[25]	1	0	0	1	—
	DONE2 (I/O)	PC[17]	1	1	1	0	VDD

Table 18-13 describes parallel I/O register programming for port A.

Table 18-13. Parallel I/O Register Programming—Port A

Channel	Signal	Pin	PPARC	PDIRC	PODRC	PSORC	Default
IDMA3	DREQ3 (I)	PC[0]	1	0	0	0	GND
	DACK3 (O)	PC[23]	1	1	0	1	—
	DONE3 (I/O)	PC[22]	1	1	1	0	VDD

18.12 IDMA Programming Examples

These programming examples demonstrate the use of most of the different modes and configurations of the IDMA channels.

18.12.1 Memory-to-Peripheral Fly-By Mode—IDMA3

In the example in Table 18-14, IDMA3 transfers data from a memory device to a 4-byte wide peripheral, both on the 60x bus. The transfers are made by issuing 4-byte read transactions to the memory and asserting DACK so the peripheral samples the data from the bus directly. No address is dedicated for the peripheral, and no internal buffer is defined in this mode. The IDMA3 channel asserts DONE on the last read transfer of the last BD to notify the peripheral that there is no data left to transfer.

Table 18-14. Example: Memory-to-Peripheral Fly-By Mode (on 60x)–IDMA3

Important Init Values	Description
DCM[FB] = 1	Fly-by mode
DCM[LP] = x	Do not care. Transfer from memory to peripheral on the 60x bus is high priority.
DCM[DMA_WRAP] = DC	Do not care. No internal buffer is used.
DCM[ERM] = 1	Transfers from peripheral are initiated by DREQ.
DCM[DT] = 1	Assertion of DONE by the peripheral terminates the transfer, interrupt EDN is set to the core, Current BD is closed and the next BD if valid is opened. Additional DREQ assertions cause the new BD to be transferred.
DCM[S/D] = 01	Memory-to-peripheral mode. DONE, DREQ, and DACK are connected to the peripheral.
DCM[SINC] = 1	The memory address is incremented after every transfer.
DCM[DINC] = 1	The memory address is incremented after every transfer.
DPR_BUF	The IDMA transfer buffer is not used.

Table 18-14. Example: Memory-to-Peripheral Fly-By Mode (on 60x)–IDMA3 (continued)

Important Init Values	Description
IBASE = IBDPTR = 0x0030	The current BD pointer is set to the BD table base address (aligned 16 -bits[3–0]=0000).
STS = 0x0004	Transfers from memory to peripheral are always 4 bytes long (60x singles).
DTS = 0x0004	Transfers from memory to peripheral are always 4 bytes long (60x singles).
Every BD[SDTB] = 0	Memory and peripheral are on the 60x bus.
Every BD[DDTB] = 0	Memory and peripheral are on the 60x bus.
Last BD[SDN] = 1	DONE is asserted on the last transfer.
Last BD[DDN] = 1	DONE is asserted on the last transfer.
IDMR3 = 0x0400_0000	The IDMA3 mask register is programmed to enable the IDSR[OB] interrupt only.
SIMR_L = 0x0000_0100	The interrupt controller is programmed to enable interrupts from IDMA3.

PDIRA = 0x2000_0000
PPARA = 0xE000_0000
PSORA = 0xE000_0000
PODRA = 0x4000_0000
	Parallel I/O registers are programmed to enable:PA[0] = DREQ3; PA[2] = DACK3; PA[1] = DONE3.
	The peripheral signals are to be connected to these lines accordingly.

RCCR = 0x0000_0080	IDMA3 configuration: DREQ is level high. DONE is high to low. request priority is higher than the SCCs.
89FE = 0x0300	IDMA3_BASE points to 0x0300 where the parameter table base address is located for IDMA3.
CPCR = 0x26C1_0009	START_IDMA command. IDMA3 page-01001 SBC-10110 op-1001 FLG=1.This write starts the channel operation.

DMA operation:
START_IDMA: Initialize all parameter RAM values, wait for DREQ to open the first BD.
Steady state: Every DREQ triggers a 4-byte transfer in single address transaction. DMA performs a memory read transaction combined with DACK assertion. Memory address is incremented constantly. Last transaction of the last BD is combined with DONE assertion.Another DREQ assertion after last BD complete will issue IDSR[OB] interrupt to the core.
STOP_IDMA: BD is closed. SC bit is set in IDSR (SC interrupt to the core is not enabled).Channel is stopped until the START_IDMA command is issued.
DONE assertion by the peripheral: current BD is closed. IDSR[EDN] is set (but the interrupt to the core is not enabled).The next BD is open with the next DREQ assertion (or IDSR[OB] interrupt is set to the core if there is no other valid BDs).

18.12.2 Memory-to-Memory (PCI Bus to 60x Bus)—IDMA2

In the example in Table 18-15, the IDMA2 channel transfers data from a memory device on the PCI bus to one on the 60x bus. IDMA2 channel reads from the PCI bus into the internal buffer in one transfer and writes to the 60x bus in nine consequent transfers of seven bursts each. It does this operation constantly by using the internal request mode. The internal buffer is set to 2 Kbytes to maximize use of the PCI bus, which is not too loaded.

The transfers to memory on the 60x bus are shorter, arbitration priority is low, and the internal request priority of IDMA2 is lowest to prevent other device starvation on the 60x bus, which is loaded.

Table 18-15. Programming Example: Memory-to-Memory (PCI-to-60x)—IDMA2

Important Init Values	Description
DCM[FB] = 0	Not in fly-by mode.
DCM[LP] = 1	Transfers to 60x have low priority; the destination bus is loaded by higher priority devices.
DCM[DMA_WRAP] = 101	The internal buffer is 2,048 bytes long. Transfers from memory on the source bus (PCI) can be as long as possible (PCI has high arbitration latency and long bursts). Transfers on the destination side are shorter, as defined in DTS.
DCM[ERM] = 0	IDMA transfers data continuously until a IDMA_STOP command is issued or until the transfer completes. DREQ, DONE, and DACK are not connected externally.
DCM[DT] = DC	Do not care. DONE assertion is not defined in memory-to-memory mode.
DCM[S/D] = 00	Memory-to-memory mode.
DCM[SINC] = 1	The source memory address is incremented after transfers.
DCM[DINC] = 1	The destination memory address is incremented after every transfer.
IBASE=IBDPTR= 0x0030	The current BD pointer is set to the BD ring base address (aligned 16- bits[3–0]=0000).
DPR_BUF = 0x0800	Initiated to address aligned to 2048 (bits[10–0] = 000_0000_0000)
SS_MAX = 2016(0x07e0)	Initiated to (internal buffer size - 32) equal to STS in this mode
STS = 2016(0x07e0)	Transfers from memory on PCI are 2016 bytes long on steady state of work.
DTS = 7×32 (0x00e0)	Transfers to memory on 60x are 224 bytes long (seven 60x bursts) for steady-state operations. We have low arbitration priority on the bus (LP = 1) but once we get it we would like to use it for more that one burst.
Every BD[SDTB] = 1	Source memory is on the PCI bus.
Every BD[DDTB] = 0	Destination memory is on the 60x bus.
Last BD[SDN] = 0	DONE is not asserted on the last transfer from memory on PCI.
Last BD[DDN] = 0	DONE is not asserted on the last transfer to memory on the 60x bus.
Last BD[L] = 1	IDMA2 is stopped after last BD complete until START_IDMA command is reissued.
Last BD[I]] = 0	IDMA2 set BC interrupt to the core after last BD complete
IDMR2=0x0f000000	IDMA2 Mask register is programmed to enable all interrupts.
SIMR_L=0x00000200	Interrupt controller is programmed to enable interrupts from IDMA2.
RCCR=0x00020000	IDMA2 configuration: Internal request priority is the lowest.
87FE=0x0200	IDMA2_BASE points to 0x0200 where the parameter table base address is located for IDMA2.

Table 18-15. Programming Example: Memory-to-Memory (PCI-to-60x)—IDMA2 (continued)

Important Init Values	Description
CPCR=0x1e810009	IDMA_start command. IDMA2 page-00111 SBC-10100 op-1001 FLG=1. This write starts the channel operation.

DMA operation:
START_IDMA: Initialize all parameter RAM values, start transferring data immediately from the first BD. Data is read from PCI bus in single burst of (32 alignment + 2016) bytes long, to fill internal buffer.The first write transfer to the 60x bus is (32 alignment + 6 or 7 burst) bytes long. The rest 8 write transfers are 7 burst long each.
Steady state: Internal buffer is filled in large burst from PCI (STS long). Data is transferred to the 60x bus in 9 transfers of 7 burst long (DTS) each. Addresses are incremented constantly. Operation continues until the last BD is completed or STOP_IDMA command is issued. When the last valid BD is complete BC interrupt is set to the core, and IDMA2 channel is stopped, until START_IDMA command is issued.
STOP_IDMA: after all data in internal buffer is written to the 60x bus, BD is closed and SC interrupt is set. Channel is stopped until START_IDMA command is issued.

Chapter 19
Serial Communications Controllers (SCCs)

The MPC8272 has four serial communications controllers (SCCs), which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks. An SCC has many physical interface options such as interfacing to TDM buses, ISDN buses, and standard modem interfaces.

The SCCs are independent from the physical interface, but SCC logic formats and manipulates data from the physical interface. Furthermore, the choice of protocol is independent from the choice of interface. An SCC is described in terms of the protocol it runs. When an SCC is programmed to a certain protocol or mode, it implements functionality that corresponds to parts of the protocol's link layer (layer 2 of the OSI reference model). Many SCC functions are common to protocols of the following controllers:

- UART, described in Chapter 20, "SCC UART Mode"
- HDLC and HDLC bus, described in Chapter 21, "SCC HDLC Mode"
- AppleTalk/LocalTalk, described in Chapter 25, "SCC AppleTalk Mode"
- BISYNC, described in Chapter 22, "SCC BISYNC Mode"
- Transparent, described in Chapter 23, "SCC Transparent Mode"
- Ethernet, described in Chapter 24, "SCC Ethernet Mode"

Although the selected protocol usually applies both to the SCC transmitter and receiver, one half of an SCC can run transparent operations while the other runs a standard protocol (except Ethernet).

Each Rx and Tx internal clock can be programmed with either an external or internal source. Internal clocks originate from one of eight baud rate generators (BRGs) or an external clock pin; see Section 15.3, "NMSI Configuration," for each SCC's available clock sources. These clocks can be as fast as a 1:4 ratio of the CPM frequency. (For example, an SCC internal clock can run at 12.5 MHz in a 50-MHz system.) However, an SCC's ability to support a sustained bit stream depends on the protocol as well as other factors.

Associated with each SCC is a digital phase-locked loop (DPLL) for external clock recovery that supports NRZ, NRZI, FM0, FM1, Manchester, and differential Manchester. If the clock recovery function is not required (that is, synchronous communication), the DPLL can be disabled, in which case only NRZ and NRZI are supported.

An SCC can be connected to its own set of pins on the MPC8272. This configuration is called the non-multiplexed serial interface (NMSI) and is described in Chapter 14, "Serial Interface with Time-Slot Assigner." Using NMSI, an SCC can support the standard modem interface signals, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. If required, software and additional parallel I/O lines can be used to support additional handshake signals. Figure 19-1 shows the SCC block diagram.

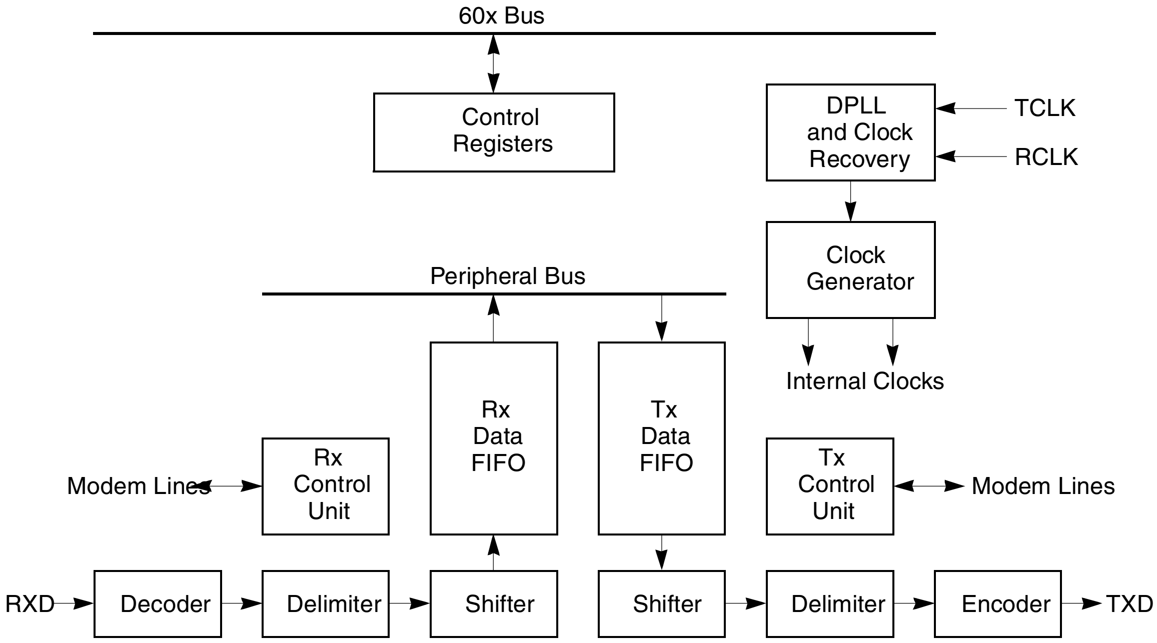


Figure 19-1. SCC Block Diagram

19.1 Features

The following is a list of the main SCC features. (Performance figures assume a 25-MHz system clock.)

- Implements HDLC/SDLC, HDLC bus, synchronous start/stop, asynchronous start/stop (UART), AppleTalk/LocalTalk, and totally transparent protocols
- Supports 10-Mbps Ethernet/IEEE 802.3 (half- or full-duplex) on all SCCs
- Additional protocols may be available through the use of RAM-based microcodes. Please contact your Freescale sales office.
- DPLL circuitry for clock recovery with NRZ, NRZI, FM0, FM1, Manchester, and differential Manchester (also known as differential bi-phase-L)
- Clocks can be derived from a baud rate generator, an external pin, or a DPLL.
- Data rate for asynchronous communication can be as high as 16.62 Mbps at 133 MHz
- Supports automatic control of the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem signals
- Multi-buffer data structure for receive and send (the number of buffer descriptors (BDs) is limited only by the size of the internal dual-port RAM—8 bytes per BD)
- Deep FIFOs (SCC transmit and receive FIFOs are 32 bytes each.)
- Transmit-on-demand feature decreases time to frame transmission (transmit latency)
- Low FIFO latency option for send and receive in character-oriented and totally transparent protocols
- Frame preamble options
- Full-duplex operation

- Fully transparent option for one half of an SCC (Rx/Tx) while another protocol executes on the other half (Tx/Rx)
- Echo and local loopback modes for testing

19.1.1 The General SCC Mode Registers (GSMR1–GSMR4)

Each SCC contains a general SCC mode register (GSMR) that defines options common to most of the protocols. GSMR_L contains the low-order 32 bits; GSMR_H, shown in Figure 19-2, contains the high-order 32 bits. Some GSMR operations are described in later sections.

	0	14	15
Field	—		
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x11A04 (GSMR1); 0x11A44 (GSMR3); 0x11A64 (GSMR4)		

	16 17	18	19	20	21	22	23	24	25	26	27	28 29	30	31
Field	TCRC	REVD	TRX	TTX	CDP	CTSP	CDS	CTSS	TFL	RFW	TXSY	SYNL	RTSM	RSYN
Reset	0000_0000_0000_0000													
R/W	R/W													
Addr	0x11A06 (GSMR1); 0x11A46 (GSMR3); 0x11A66 (GSMR4)													

Figure 19-2. GSMR_H—General SCC Mode Register (High Order)

Table 19-1 describes GSMR_H fields.

Table 19-1. GSMR_H Field Descriptions

Bit	Name	Description
0–15	—	Reserved, should be cleared
16–17	TCRC	Transparent CRC (valid for totally transparent channel only). Selects the frame checking provided on transparent channels of the SCC (either the receiver, transmitter, or both, as defined by TTX and TRX). Although this configuration selects a frame check type, the decision to send the frame check is made in the TxBD. Thus, frame checks are not needed in transparent mode and frame check errors generated on the receiver can be ignored. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1). 01 CRC16 (BISYNC). (X16 + X15 + X2 + 1). 10 32-bit CCITT CRC (Ethernet and HDLC). (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 + 1). 11 Reserved
18	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 Reverses the bit order for totally transparent channels on this SCC (either the receiver, transmitter, or both) and sends the msb of each byte first. Section 22.11, “BISYNC Mode Register (PSMR),” describes reversing bit order in a BISYNC protocol.

Table 19-1. GSMR_H Field Descriptions (continued)

Bit	Name	Description
19–20	TRX, TTX	Transparent receiver/transmitter. The receiver, transmitter, or both can use totally transparent operation, regardless of GSMR_L[MODE]. For example, to configure the transmitter as a UART and the receiver for totally transparent operations, set MODE = 0b0100 (UART), TTX = 0, and TRX = 1. 0 Normal operation 1 The channel uses totally transparent mode, regardless of the protocol chosen in GSMR_L[MODE]. For full-duplex totally transparent operation, set both TTX and TRX. **Note:** An SCC cannot operate with half in Ethernet mode and half in transparent mode. That is, if MODE = 0b1100 (Ethernet), erratic operation occurs unless TTX = TRX.
21, 22	CDP, CTSP	$\overline{\text{CD}}$/$\overline{\text{CTS}}$ pulse. If this SCC is used in the TSA and is programmed in transparent mode, set CTSP and refer to Section 23.4.2, "Synchronization and the TSA," for options on programming CDP. 0 Normal operation (envelope mode). $\overline{\text{CD}}$/$\overline{\text{CTS}}$ should envelope the frame. Negating $\overline{\text{CD}}$/$\overline{\text{CTS}}$ during reception causes a $\overline{\text{CD}}$/$\overline{\text{CTS}}$ lost error. 1 Pulse mode. Synchronization occurs when $\overline{\text{CD}}$/$\overline{\text{CTS}}$ is asserted; further $\overline{\text{CD}}$/$\overline{\text{CTS}}$ transitions do not affect reception.
23, 24	CDS, CTSS	$\overline{\text{CD}}$/$\overline{\text{CTS}}$ sampling. Determine synchronization characteristics of $\overline{\text{CD}}$ and $\overline{\text{CTS}}$. If the SCC is in transparent mode and is used in the TSA, CDS and CTSS must be set. Also, CDS and CTSS must be set for loopback testing in transparent mode. 0 $\overline{\text{CD}}$/$\overline{\text{CTS}}$ is assumed to be asynchronous with data. It is internally synchronized by the SCC, then data is received ($\overline{\text{CD}}$) or sent ($\overline{\text{CTS}}$) after several clock delays. 1 $\overline{\text{CD}}$/$\overline{\text{CTS}}$ is assumed to be synchronous with data, which speeds up operation. $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ must transition while the Rx/Tx clock is low, at which time, the transfer begins. Useful for connecting MPC8272 in transparent mode since the $\overline{\text{RTS}}$ of one MPC8272 can connect directly to the $\overline{\text{CD}}$/$\overline{\text{CTS}}$ of another.
25	TFL	Transmit FIFO length. 0 Normal operation. The transmit FIFO is 32 bytes. 1 The Tx FIFO is 1 byte. This option is used with character-oriented protocols, such as UART, to ensure a minimum FIFO latency at the expense of performance.
26	RFW	Rx FIFO width. 0 Receive FIFO is 32 bits wide for maximum performance; the Rx FIFO is 32 bytes. Data is not normally written to receive buffers until at least 32 bits are received. This configuration is required for HDLC-type protocols and Ethernet and is recommended for high-performance transparent protocols. 1 Low-latency operation. The receive FIFO is 8 bits wide, reducing the Rx FIFO to a quarter its normal size. This allows data to be written to the buffer as soon as a character is received, instead of waiting to receive 32 bits. This configuration must be chosen for character-oriented protocols, such as UART. It can also be used for low-performance, low-latency, transparent operation. However, it must not be used with HDLC, HDLC Bus, AppleTalk, or Ethernet because it causes erratic behavior.
27	TXSY	Transmitter synchronized to the receiver. Intended for X.21 applications where the transmitted data must begin an exact multiple of 8-bit periods after the received data arrives. 0 No synchronization between receiver and transmitter (default). 1 The transmit bit stream is synchronized to the receiver. Additionally, if RSYN = 1, transmission in totally transparent mode does not occur until the receiver synchronizes with the bit stream and $\overline{\text{CTS}}$ is asserted to the SCC. Assuming $\overline{\text{CTS}}$ is asserted, transmission begins 8 clocks after the receiver starts receiving data.

Table 19-1. GSMR_H Field Descriptions (continued)

Bit	Name	Description
28–29	SYNL	Sync length (BISYNC and transparent mode only). See the data synchronization register (DSR) definition in Section 22.9, “Sending and Receiving the Synchronization Sequence,” (BISYNC) and Section 23.4.1.1, “In-Line Synchronization Pattern,” (transparent). 00 An external sync ($\overline{\text{CD}}$) is used instead of the sync pattern in the DSR. 01 4-bit sync. The receiver synchronizes on a 4-bit sync pattern stored in the DSR. This sync and additional syncs can be stripped by programming the SCC’s parameter RAM for character recognition. 10 8-bit sync. Should be chosen along with the BISYNC protocol to implement mono-sync. The receiver synchronizes on an 8-bit sync pattern in the DSR. 11 16-bit sync. Also called BISYNC. The receiver synchronizes on a 16-bit sync pattern stored in the DSR.
30	RTSM	$\overline{\text{RTS}}$ mode. Determines whether flags or idles are to be sent. Can be changed on-the-fly. 0 Send idles between frames as defined by the protocol and the TEND bit. $\overline{\text{RTS}}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{\text{RTS}}$ is always asserted whenever the SCC is enabled.
31	RSYN	Receive synchronization timing (totally transparent mode only). 0 Normal operation 1 If CDS = 1, $\overline{\text{CD}}$ should be asserted on the second bit of the Rx frame rather than on the first.

Figure 19-3 shows GSMR_L.

	0	1	2	3	4	5	6	7	8		10	11	12	13	14	15
Field	—	EDGE		TCI	TSNC		RINV	TINV	TPL			TPP		TEND	TDCR	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11A00 (SCC1); 0x11A40 (SCC3); 0x11A60 (SCC4)															

	16	17	18		20	21		23	24	25	26	27	28			31
Field	RDCR		RENC			TENC			DIAG		ENR	ENT	MODE			
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11A02 (SCC1); 0x11A42 (SCC3); 0x11A62 (SCC4)															

Figure 19-3. GSMR_L—General SCC Mode Register (Low Order)

Table 19-2 describes GSMR_L fields.

Table 19-2. GSMR_L Field Descriptions

Bit	Name	Description
0	—	Reserved, should be cleared

Table 19-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
1–2	EDGE	Clock edge. Determines the clock edge the DPLL uses to adjust the receive sample point due to jitter in the received signal. Ignored in UART protocol or if the 1x clock mode is selected in RDCR. 00 Both the positive and negative edges are used for changing the sample point (default). 01 Positive edge. Only the positive edge of the received signal is used to change the sample point. 10 Negative edge. Only the negative edge of the received signal is used to change the sample point. 11 No adjustment is made to the sample point.
3	TCI	Transmit clock invert. 0 Normal operation 1 Before it is used, the internal Tx clock (TCLK) is inverted by the SCC so it can clock data out one-half clock earlier (on the rising rather than the falling edge). In this case, the SCC offers a minimum and maximum rising clock edge-to-data specification. Data output by the SCC after the rising edge of an external Tx clock can be latched by the external receiver one clock cycle later on the next rising edge of the same Tx clock. The edge on which the SCC outputs the data depends on the mode of operation: • In HDLC and Transparent mode, when TCI=0, data is sent on the falling edge; when TCI=1, on the rising edge. • In Ethernet mode, when TCI=0, data is sent on the rising edge; when TCI=1, on the falling edge. **Note:** Recommended for Ethernet, HDLC, and transparent operation when clock rates exceed 8 MHz to improve data setup time for the external transceiver.
4–5	TSNC	Transmit sense. Determines the amount of time the internal carrier sense signal stays active after the last transition on RXD, indicating that the line is free. For instance, AppleTalk can use TSNC to avoid a spurious CS-changed (SCCE[DCC]) interrupt that would otherwise occur during the frame sync sequence before the opening flags. If RDCR is configured to 1× clock mode, the delay is the greater of the two numbers listed. If RDCR is configured to 8×, 16×, or 32× mode, the delay is the smaller number. 00 Infinite. Carrier sense is always active (default). 01 14- or 6.5-bit times as determined by RDCR 10 4- or 1.5-bit times as determined by RDCR (normally for AppleTalk) 11 3- or 1-bit times as determined by RDCR
6	RINV	DPLL Rx input invert data. Must be zero in HDLC bus mode or asynchronous UART mode. 0 Do not invert. 1 Invert data before sending it to the DPLL for reception. Used to produce FM1 from FM0 and NRZI space from NRZI mark or to invert the data stream in regular NRZ mode.
7	TINV	DPLL Tx input invert data. Must be zero in HDLC bus mode. 0 Do not invert. 1 Invert data before sending it to the DPLL for transmission. Used to produce FM1 from FM0 and NRZI space from NRZI mark and to invert the data stream in regular NRZ mode. In T1 applications, setting TINV and TEND creates a continuously inverted HDLC data stream.
8–10	TPL	Tx preamble length. Determines the length of the preamble configured by the TPP bits. 000 No preamble (default). 001 8 bits (1 byte) 010 16 bits (2 bytes) 011 32 bits (4 bytes) 100 48 bits (6 bytes). Select this setting for Ethernet operation. 101 64 bits (8 bytes) 110 128 bits (16 bytes) 111 Reserved

Table 19-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
11–12	TPP	Tx preamble pattern. Determines what, if any, bit pattern should precede each Tx frame. The preamble pattern is sent before the first flag/sync of the frame. TPP is ignored in UART mode. The preamble length is programmed in TPL; the preamble pattern is typically sent to a receiving station that uses a DPLL for clock recovery. The receiving DPLL uses the regular preamble pattern to help it lock onto the received signal in a short, predictable time period. 00 All zeros 01 Repetitive 10s. Select this setting for Ethernet operation. 10 Repetitive 01s 11 All ones. Select this setting for LocalTalk operation.
13	TEND	Transmitter frame ending. Intended for NRZI transmitter encoding of the DPLL. TEND determines whether TXD should idle in a high state or in an encoded ones state (high or low). It can, however, be used with other encodings besides NRZI. 0 Default operation. TXD is encoded only when data is sent, including the preamble and opening and closing flags/syncs. When no data is available to send, the signal is driven high. 1 TXD is always encoded, even when idles are sent.
14–15	TDCR	Transmitter/receiver DPLL clock rate. If the DPLL is not used, choose 1× mode except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. TDCR should match RDCR in most applications to allow the transmitter and receiver to use the same clock source. If an application uses the DPLL, the selection of TDCR/RDCR depends on the encoding/decoding. If communication is synchronous, select 1×. FM0/FM1, Manchester, and Differential Manchester require 8×, 16×, or 32×. If NRZ- or NRZI-encoded communication is asynchronous (that is, clock recovery required), select 8×, 16×, or 32×. The 8× option allows highest speed, whereas the 32× option provides the greatest resolution. 00 1× clock mode. Only NRZ or NRZI encodings/decodings are allowed. 01 8× clock mode 10 16× clock mode. Normally chosen for UART and AppleTalk. 11 32× clock mode
16–17	RDCR	
18–20	RENC	Receiver decoding/transmitter encoding method. Select NRZ if DPLL is not used. RENC should equal TENC in most applications. However, do not use this internal DPLL for Ethernet. 000 NRZ (default setting if DPLL is not used). Required for UART (synchronous or asynchronous). 001 NRZI Mark (set RINV/TINV also for NRZI space) 010 FM0 (set RINV/TINV also for FM1) 011 Reserved 100 Manchester 101 Reserved 110 Differential Manchester (differential bi-phase-L) 111 Reserved
21–23	TENC	

Table 19-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
24–25	DIAG	Diagnostic mode. 00 Normal operation, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are under automatic control. Data is received through RXD and transmitted through TXD. The SCC uses modem signals to enable or disable transmission and reception. These timings are shown in Section 19.3.5, “Controlling SCC Timing with RTS, CTS, and CD.” 01 Local loopback mode. Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. The value on RXD is ignored. If enabled, data appears on TXD, or the parallel I/O registers can be programmed to make TXD high. $\overline{\text{RTS}}$ can also be programmed to be disabled in the appropriate parallel I/O register. The transmitter and receiver must share the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{\text{RTS}}$ connected to $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ grounded) or set the parallel I/O registers so $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are permanently asserted to the SCC by configuring the associated $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins as general-purpose I/O. 10 Automatic echo mode. The transmitter automatically resends received data bit-by-bit using the Rx clock provided. The receiver operates normally and receives data if $\overline{\text{CD}}$ is asserted. $\overline{\text{CTS}}$ is ignored. 11 Loopback and echo mode. Loopback and echo operation occur simultaneously. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are ignored. See the loopback bit description above for clocking requirements. For TDM operation, the diagnostic mode is selected by SI2MR[SDM*x*]; see Section 14.5.2, “SI2 Mode Register (SI2MR).”
26	ENR	Enable receive. Enables the receiver hardware state machine for this SCC. 0 The receiver is disabled and data in the Rx FIFO is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR can be set or cleared, regardless of whether serial clocks are present. Section 19.3.7, “Reconfiguring the SCCs,” describes how to disable/enable an SCC. Note that other tools, including the ENTER HUNT MODE and CLOSE RXBD commands and the E bit of the Rx BD, data provide the capability to control the receiver.
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this SCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the current character is aborted and TXD returns to the idle state. Data already in the Tx shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. Section 19.3.7, “Reconfiguring the SCCs,” describes how to disable/enable an SCC. Note that other tools, such as the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, the freeze option and $\overline{\text{CTS}}$ flow control option in UART mode, and the R bit of the TxBD, also provide the capability to control the transmitter.

Table 19-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
28–31	MODE	Channel protocol mode. See also GSMR_H[TTX, TRX]. 0000 HDLC 0001 Reserved 0010 AppleTalk/LocalTalk 0011 SS7—reserved for RAM microcode 0100 UART 0101 Profibus—reserved for RAM microcode 0110 Reserved 0111 Reserved 1000 BISYNC 1001 Reserved 1010 QMC 1011 Reserved 1100 Ethernet 11xx Reserved

19.1.2 Protocol-Specific Mode Register (PSMR)

The protocol implemented by an SCC is selected by its GSMR_L[MODE]. Each SCC has an additional protocol-specific mode register (PSMR) that configures it specifically for the chosen protocol. The PSMR fields are described in the specific chapters that describe each protocol. PSMRs are cleared at reset. PSMRs reside at the following addresses: 0x11A08 (PSMR1), 0x11A48 (PSMR3), and 0x11A68 (PSMR4).

19.1.3 Data Synchronization Register (DSR)

Each SCC has a data synchronization register (DSR) that specifies the pattern used for frame synchronization. The programmed value for DSR depends on the protocol:

- UART—DSR is used to configure fractional stop bit transmission.
- BISYNC and transparent—DSR should be programmed with the sync pattern.
- Ethernet—DSR should be programmed with 0xD555.
- HDLC—At reset, DSR defaults to 0x7E7E (two HDLC flags), so it does not need to be written.

Figure 19-4 shows the sync fields.

	0 7	8 15
Field	SYN2	SYN1
Reset	0111_1110	0111_1110
R/W	R/W	
Addr	0x11A0E (DSR1); 0x11A4E (DSR3); 0x11A6E (DSR4)	

Figure 19-4. Data Synchronization Register (DSR)

19.1.4 Transmit-on-Demand Register (TODR)

In normal operation, if no frame is being sent by an SCC, the CP periodically polls the R bit of the next TxBD to see if a new frame/buffer is requested. Depending on the SCC configuration, this polling occurs every 8–32 serial Tx clocks. The transmit-on-demand option, selected in the transmit-on-demand register (TODR) shown in Figure 19-5, shortens the latency of the Tx buffer/frame and is useful in LAN-type protocols where maximum inter-frame gap times are limited by the protocol specification.

	0	15
Field	TOD	—
Reset	0000_0000_0000_0000	
R/W	W	
Addr	0x11A0C (TODR1); 0x11A4C (TODR3); 0x11A6C (TODR4)	

Figure 19-5. Transmit-on-Demand Register (TODR)

The CP can be configured to begin processing a new frame/buffer without waiting the normal polling time by setting TODR[TOD] after TxBD[R] is set. Because this feature favors the specified TxBD, it may affect servicing of other SCC FIFOs. Therefore, transmitting on demand should only be used when a high-priority TxBD has been prepared and enough time has passed since the last transmission. Table 19-3 describes TODR fields.

Table 19-3. TODR Field Descriptions

Bits	Name	Description
0	TOD	Transmit on demand. 0 Normal operation 1 The CP gives high priority to the current TxBD and begins sending the frame without waiting the normal polling time to check the TxBD's R bit. TOD is cleared automatically after one serial clock, but transmitting on demand continues until an unprepared (R = 0) BD is reached. TOD does not need to be set again if new TxBDs are added to the BD table as long as older TxBDs are still being processed. New TxBDs are processed in order. The first bit of the frame is typically clocked out 5–6 bit times after TOD is set.
1–15	—	Reserved, should be cleared

19.2 SCC Buffer Descriptors (BDs)

Data associated with each SCC channel is stored in buffers and each buffer is referenced by a buffer descriptor (BD) that can reside anywhere in dual-port RAM. The total number of 8-byte BDs is limited only by the size of the dual-port RAM (128 BDs/1 Kbyte). These BDs are shared among all serial controllers—SCCs, SMCs, SPI, and I^{2}C. The user defines how the BDs are allocated among the controllers.

Each 64-bit BD has the following structure:

- The half word at offset + 0x0 contains status and control bits that control and report on the data transfer. These bits vary from protocol to protocol. The CPM updates the status bits after the buffer is sent or received.
- The half word at offset + 0x2 (data length) holds the number of bytes sent or received.

- For an RxBD, this is the number of bytes the controller writes into the buffer. The CPM writes the length after received data is placed into the associated buffer and the buffer closed. In frame-based protocols (but not including SCC transparent operation), this field contains the total frame length, including CRC bytes. Also, if a received frame’s length, including CRC, is an exact multiple of MRBLR, the last BD holds no actual data but does contain the total frame length.
- For a TxBD, this is the number of bytes the controller should send from its buffer. Normally, this value should be greater than zero. The CPM never modifies this field.

- The word at offset + 0x4 (buffer pointer) points to the beginning of the buffer in memory (internal or external).
  - For an RxBD, the value must be a multiple of four (word-aligned).
  - For a TxBD, this pointer can be even or odd.

Shown in Figure 19-6, the format of Tx and Rx BDs is the same in each SCC mode. Only the status and control bits differ for each protocol.

	0	15
Offset + 0	Status and Control	
Offset + 2	Data Length	
Offset + 4	High-Order Buffer Pointer	
Offset + 6	Low-Order Buffer Pointer	

Figure 19-6. SCC Buffer Descriptors (BDs)

For frame-oriented protocols, a message can reside in as many buffers as necessary. Each buffer has a maximum length of 65,535 bytes. The CPM does not assume that all buffers of a single frame are currently linked to the BD table. The CPM does assume, however, that the unlinked buffers are provided by the core in time to be sent or received; otherwise, an error condition is reported—an underrun error when sending and a busy error when receiving. Figure 19-7 shows the SCC BD table and buffer structure.

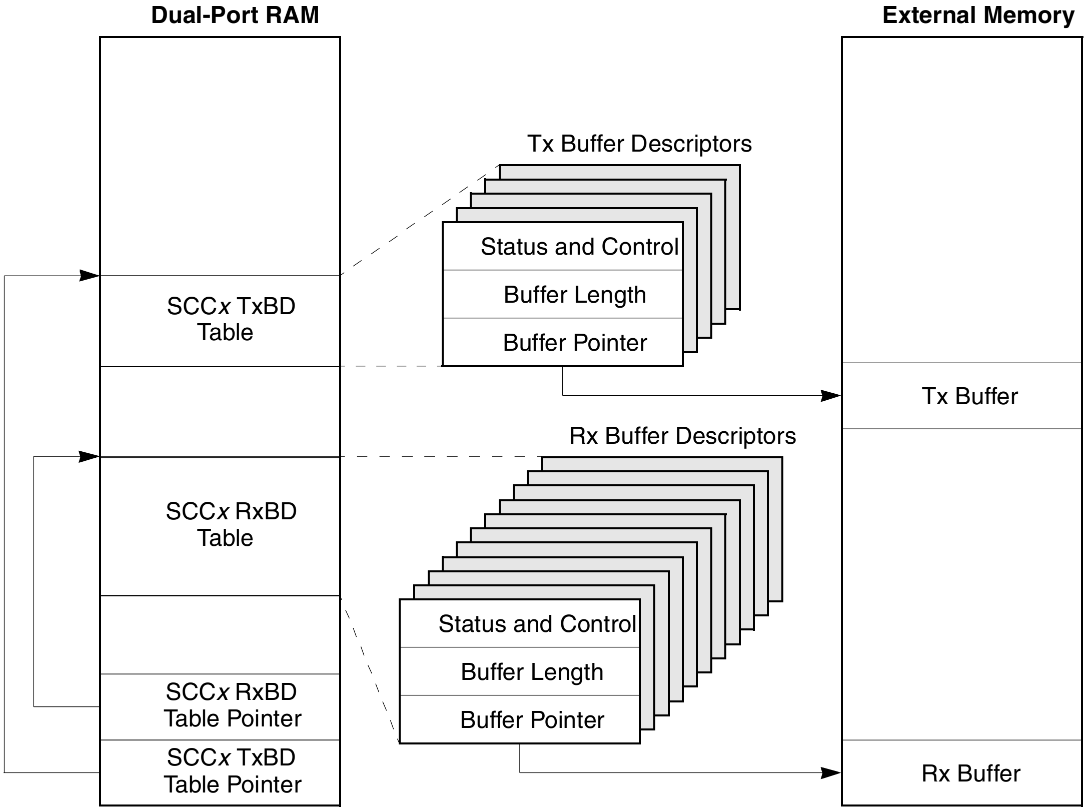


Figure 19-7. SCC BD and Buffer Memory Structure

In all protocols, BDs can point to buffers in the internal dual-port RAM. However, because dual-port RAM is used for descriptors, buffers are usually put in external RAM, especially if they are large.

The CPM processes TxBDs straightforwardly; when the transmit side of an SCC is enabled, the CPM starts with the first BD in that SCC TxBD table. Once the CPM detects that the R bit is set in the TxBD, it starts processing the buffer. The CPM detects that the BD is ready when it polls the R bit or when the user writes to the TODR. After data from the BD is put in the Tx FIFO, if necessary the CPM waits for the next descriptor's R bit to be set before proceeding. Thus, the CPM does no look-ahead descriptor processing and does not skip BDs that are not ready. When the CPM sees a BD's W bit (wrap) set, it returns to the start of the BD table after this last BD of the table is processed. The CPM clears R (not ready) after using a TxBD, which keeps it from being retransmitted before it is confirmed by the core. However, some protocols support a continuous mode (CM), for which R is not cleared (always ready).

The CPM uses RxBDs similarly. When data arrives, the CPM performs required processing on the data and moves resultant data to the buffer pointed to by the first BD; it continues until the buffer is full or an event, such as an error or end-of-frame detection, occurs. The buffer is then closed; subsequent data uses the next BD. If E = 0, the current buffer is not empty and it reports a busy error. The CPM does not move from the current BD until E is set by the core (the buffer is empty). After using a descriptor, the CPM clears E (not empty) and does not reuse a BD until it has been processed by the core. However, in continuous mode (CM), E remains set. When the CPM discovers a descriptor's W bit set (indicating it is the last BD in the circular BD table), it returns to the beginning of the table when it is time to move to the next buffer.

19.3 SCC Parameter RAM

Each SCC parameter RAM area begins at the same offset from each SCC base area. Section 19.3.1, “SCC Base Addresses,” describes the SCC’s base addresses. The protocol-specific portions of the SCC parameter RAM are discussed in the specific protocol descriptions and the part that is common to all SCC protocols is shown in Table 19-4.

Some parameter RAM values must be initialized before the SCC can be enabled. Other values are initialized or written by the CPM. Once initialized, most parameter RAM values do not need to be accessed because most activity centers around the descriptors rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.
- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 19.3.7, “Reconfiguring the SCCs.”

Table 19-4 shows the parameter RAM map for all SCC protocols. Boldfaced entries must be initialized by the user.

Table 19-4. SCC Parameter RAM Map for All Protocols

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address—offset from the beginning of dual-port RAM. The BD tables can be placed in any unused portion of the dual-port RAM. The CPM starts BD processing at the top of the table. (The user defines the end of the BD table by setting the W bit in the last BD to be processed.) Initialize these entries before enabling the corresponding channel. Erratic operations occur if BD tables of active SCCs overlap. Values in RBASE and TBASE should be multiples of eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx function code. See Section 19.3.2, “Function Code Registers (RFCR and TFCR).”
0x05	**TFCR**	Byte	Tx function code. See Section 19.3.2, “Function Code Registers (RFCR and TFCR).”
0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8272 writes to a receive buffer before it goes to the next buffer. The MPC8272 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 (GSMR_H[RFW] = 1). **Note:** Although MRBLR is not intended to be changed while the SCC is operating, it can be changed dynamically in a single-cycle, 16-bit move (not two 8-bit cycles). Changing MRBLR has no immediate effect. To guarantee the exact 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.

Table 19-4. SCC Parameter RAM Map for All Protocols (continued)

Offset [1]	Name	Width	Description
0x08	RSTATE	Word	Rx internal state[3]
0x0C		Word	Rx internal buffer pointer[2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	Current RxBD pointer. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes RBPTR to the value in the RBASE. Although most applications do not need to write RBPTR, it can be modified when the receiver is disabled or when no Rx buffer is in use.
0x12		Hword	Rx internal byte count [2]. The Rx internal byte count is a down-count value initialized with MRBLR and decremented with each byte written by the supporting SDMA channel.
0x14		Word	Rx temp[3]
0x18	TSTATE	Word	Tx internal state[3]
0x1C		Word	Tx internal buffer pointer [2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	Current TxBD pointer. Points to the current BD being processed or to the next BD the transmitter uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes TBPTR to the value in the TBASE. Although most applications do not need to write TBPTR, it can be modified when the transmitter is disabled or when no Tx buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission).
0x22		Hword	Tx internal byte count [2]. A down-count value initialized with TxBD[Data Length] and decremented with each byte read by the supporting SDMA channel.
0x24		Word	Tx temp[3]
0x28	RCRC	Word	Temp receive CRC [2]
0x2C	TCRC	Word	Temp transmit CRC [2]
0x30			Protocol-specific area. (The size of this area depends on the protocol chosen.)

[1] From SCC base. See Section 19.3.1, “SCC Base Addresses.”

[2] These parameters need not be accessed for normal operation but may be helpful for debugging.

[3] For CP use only.

19.3.1 SCC Base Addresses

The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SMCs, SPI, I^2C, and IDMA channels. SCC base addresses are described in Table 19-5.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM. For example, the Ethernet parameter RAM is defined differently in some locations from the HDLC-specific parameter RAM.

Table 19-5. Parameter RAM—SCC Base Addresses

Page	Address [1]	Peripheral	Size (Bytes)
1	0x8000	SCC1	256
2	0x8100	Reserved	256
3	0x8200	SCC3	256
4	0x8300	SCC4	256

[1] Offset from RAM_Base.

19.3.2 Function Code Registers (RFCR and TFCR)

There are six separate function code registers for the three SCC channels, three for Rx buffers (RFCR1–RFCR4) and three for Tx buffers (TFCR1–TFCR4). The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 19-8 shows the register format.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**TC2**	**DTB**	—
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	SCC*x* base + 0x04 (RFCR*x*); SCC*x* base + 0x05 (TFCR*x*)							

Figure 19-8. Function Code Registers (RFCR and TFCR)

Table 19-6 describes RFCR*x*/TFCR*x* fields.

Table 19-6. RFCR*x* /TFCR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator 0 Use 60x bus for SDMA operation 1 Reserved
7	—	Reserved, should be cleared

19.3.3 Handling SCC Interrupts

To allow interrupt handling for SCC-specific events, event, mask, and status registers are provided within each SCC's internal memory map area; see Table 19-7. Because interrupt events are protocol-dependent, event descriptions are found in the specific protocol chapters.

Table 19-7. SCC*x* Event, Mask, and Status Registers

Register & IMMR Offset	Description

SCCE*x*
0x11A10 (SCCE1);
0x11A50 (SCCE3);
0x11A70 (SCCE4)
	SCC event register. This 16-bit register reports events recognized by any of the SCCs. When an event is recognized, the SCC sets its corresponding bit in SCCE, regardless of the corresponding mask bit. When the corresponding event occurs, an interrupt is signaled to the SIVEC register. Bits are cleared by writing ones (writing zeros has no effect). SCCE is cleared at reset and can be read at any time.

SCCM*x*
0x11A14 (SCCM1);
0x11A54 (SCCM3);
0x11A74 (SCCM4)
	SCC mask register. The 16-bit, read/write register allows interrupts to be enabled or disabled using the CPM for specific events in each SCC channel. An interrupt is generated only if SCC interrupts in this channel are enabled in the SIU interrupt mask register (SIMR). If an SCCM bit is zero, the CPM does not proceed with interrupt handling when that event occurs. The SCCM and SCCE bit positions are identical.

SCCS*x*
0x11A17 (SCCS1);
0x11A57 (SCCS3);
0x11A77 (SCCS4)
	SCC status register. This 8-bit, read-only register allows monitoring of the real-time status of RXD.

Follow these steps to handle an SCC interrupt:

1. When an interrupt occurs, read SCCE to determine the interrupt sources and clear those SCCE bits (in most cases).
2. Process the TxBDs to reuse them if SCCE[TX] or SCCE[TXE] = 1. If the transmit speed is fast or the interrupt delay is long, the SCC may have sent more than one Tx buffer. Thus, it is important to check more than one TxBD during interrupt handling. A common practice is to process all TxBDs in the handler until one is found with its R bit set.
3. Extract data from the RxBD if SCCE[RX], SCCE[RXB], or SCCE[RXF] is set. As with transmit buffers, if the receive speed is fast or the interrupt delay is long, the SCC may have received more than one buffer and the handler should check more than one RxBD. A common practice is to process all RxBDs in the interrupt handler until one is found with RxBD[E] set.
4. Execute the **rfi** instruction.

Additional information about interrupt handling can be found in Section 4.2, "Interrupt Controller."

19.3.4 Initializing the SCCs

The SCCs require that a number of registers and parameters be configured after a power-on reset. Regardless of the protocol used, follow these steps to initialize SCCs:

1. Write the parallel I/O ports to configure and connect the I/O pins to the SCCs.
2. Configure the parallel I/O registers to enable $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ if these signals are required.

3. If the time-slot assigner (TSA) is used, the serial interface (SI2) must be configured. If the SCC is used in NMSI mode, CMXSCR must still be initialized.
4. Write all GSMR bits except ENT or ENR.
5. Write the PSMR.
6. Write the DSR.
7. Initialize the required values for this SCC’s parameter RAM.
8. Initialize the transmit/receive parameters through the CP command register (CPCR).
9. Clear out any current events in SCCE (optional).
10. Write ones to SCCM register to enable interrupts.
11. Set GSMR_L[ENT] and GSMR_L[ENR].

Descriptors can have their R or E bits set at any time. Notice that the CPCR does not need to be accessed after a hardwarereset. An SCC should be disabled and reenabled after any dynamic change to its parallel I/O ports or serial channel physical interface configuration. A full reset can also be implemented using CPCR[RST].

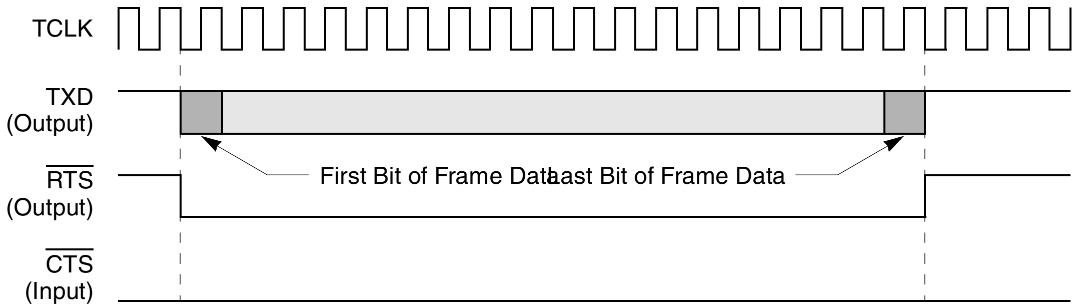
19.3.5 Controlling SCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

When GSMR_L[DIAG] is programmed to normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are controlled by the SCC. In the following subsections, it is assumed that GSMR_L[TCI] is zero, implying normal transmit clock operation.

19.3.5.1 Synchronous Protocols

$\overline{\text{RTS}}$ is asserted when the SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. At this point, the SCC starts sending data once appropriate conditions occur on $\overline{\text{CTS}}$. In all cases, the first data bit is the start of the opening flag, sync pattern, or preamble.

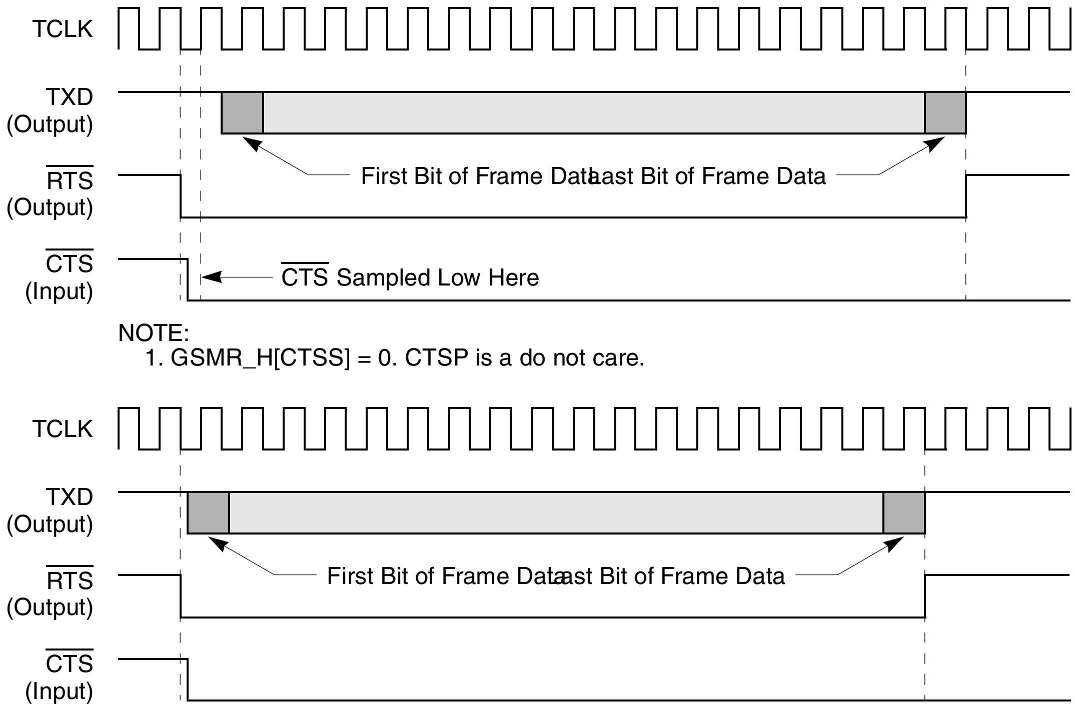
Figure 19-9 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit times, regardless of GSMR_H[CTSS]. This operation assumes that $\overline{\text{CTS}}$ is already asserted to the SCC or that $\overline{\text{CTS}}$ is reprogrammed to be a parallel I/O line, in which case $\overline{\text{CTS}}$ to the SCC is always asserted. $\overline{\text{RTS}}$ is negated one clock after the last bit in the frame.



NOTE:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 19-9. Output Delay from $\overline{\text{RTS}}$ Asserted for Synchronous Protocols

When $\overline{\text{RTS}}$ is asserted, if $\overline{\text{CTS}}$ is not already asserted, delays to the first data bit depend on when $\overline{\text{CTS}}$ is asserted. Figure 19-10 shows that the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit times or no delay, depending on GSMR_H[CTSS].

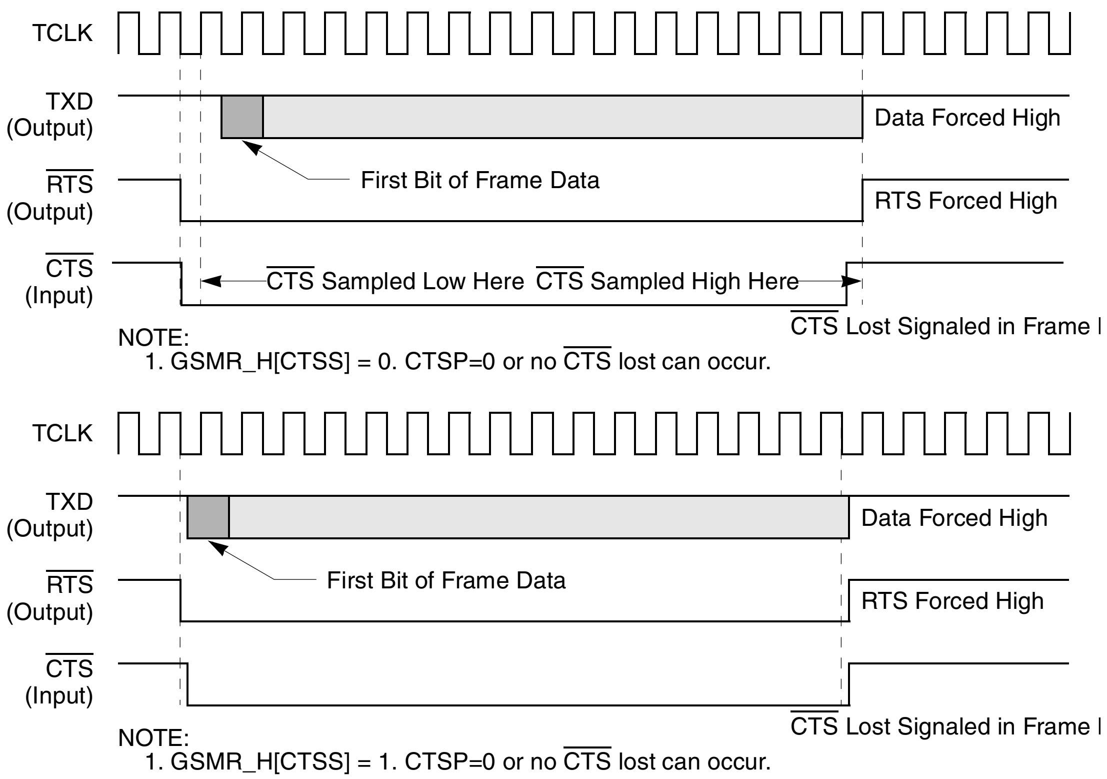


NOTE:
1. GSMR_H[CTSS] = 0. CTSP is a do not care.

NOTE:
1. GSMR_H[CTSS] = 1. CTSP is a do not care.

Figure 19-10. Output Delay from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

If $\overline{\text{CTS}}$ is programmed to envelope data, negating it during frame transmission causes a $\overline{\text{CTS}}$ lost error. Negating $\overline{\text{CTS}}$ forces $\overline{\text{RTS}}$ high and Tx data to become idle. If GSMR_H[CTSS] is zero, the SCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized; otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 19-11.


TCLK
TXD (Output)
$\overline{\text{RTS}}$ (Output)
$\overline{\text{CTS}}$ (Input)
Data Forced High
First Bit of Frame Data
RTS Forced High
$\overline{\text{CTS}}$ Sampled Low Here
$\overline{\text{CTS}}$ Sampled High Here
$\overline{\text{CTS}}$ Lost Signaled in Frame I
TCLK
TXD (Output)
$\overline{\text{RTS}}$ (Output)
$\overline{\text{CTS}}$ (Input)
Data Forced High
First Bit of Frame Data
RTS Forced High
$\overline{\text{CTS}}$ Lost Signaled in Frame I

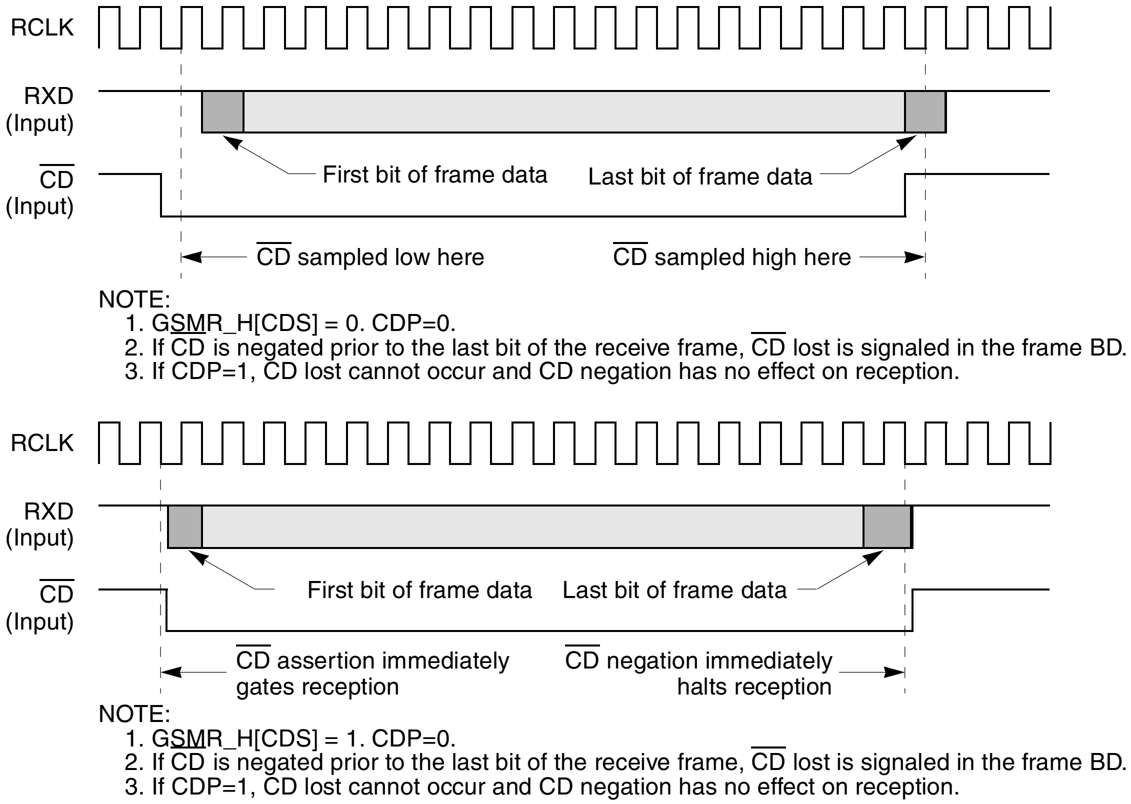
NOTE:
1. GSMR_H[CTSS] = 0. CTSP=0 or no $\overline{\text{CTS}}$ lost can occur.

NOTE:
1. GSMR_H[CTSS] = 1. CTSP=0 or no $\overline{\text{CTS}}$ lost can occur.

Figure 19-11. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

Note that if GSMR_H[CTSS] = 1, $\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as shown in Figure 19-12. If GSMR_H[CDS] is zero, $\overline{\text{CD}}$ is sampled on the rising Rx clock edge before data is received. If GSMR_H[CDS] is 1, $\overline{\text{CD}}$ transitions cause data to be immediately gated into the receiver.


RCLK
RXD
(Input)
$\overline{\text{CD}}$
(Input)
First bit of frame data
Last bit of frame data
$\overline{\text{CD}}$ sampled low here
$\overline{\text{CD}}$ sampled high here
RCLK
RXD
(Input)
$\overline{\text{CD}}$
(Input)
First bit of frame data
Last bit of frame data
$\overline{\text{CD}}$ assertion immediately gates reception
$\overline{\text{CD}}$ negation immediately halts reception

NOTE:
1. GSMR_H[CDS] = 0. CDP=0.
2. If CD is negated prior to the last bit of the receive frame, $\overline{\text{CD}}$ lost is signaled in the frame BD.
3. If CDP=1, CD lost cannot occur and CD negation has no effect on reception.

NOTE:
1. GSMR_H[CDS] = 1. CDP=0.
2. If CD is negated prior to the last bit of the receive frame, $\overline{\text{CD}}$ lost is signaled in the frame BD.
3. If CDP=1, CD lost cannot occur and CD negation has no effect on reception.

Figure 19-12. 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 GSMR_H[CDS] is zero, $\overline{\text{CD}}$ must be sampled by the SCC 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 GSMR_H[CDS] is set, all $\overline{\text{CD}}$ transitions must occur while the Rx clock is low.

19.3.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{\text{RTS}}$ is asserted when SCC 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 20, "SCC UART Mode."

- If $\overline{\text{CTS}}$ is already asserted when $\overline{\text{RTS}}$ is asserted, transmission begins in 2 additional bit times.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted and GSMR_H[CTSS] = 0, transmission begins in 3 additional bit times.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted and GSMR_H[CTSS] = 1, transmission begins in 2 additional bit times.

19.3.6 Digital Phase-Locked Loop (DPLL) Operation

Each SCC channel includes a digital phase-locked loop (DPLL) for recovering clock information from a received data stream. For applications that provide a direct clock source to the SCC, the DPLL can be bypassed by selecting 1x mode for GSMR_L[RDCR, TDCR]. If the DPLL is bypassed, only NRZ or NRZI encodings are available. The DPLL must not be used when an SCC is programmed to Ethernet and is optional for other protocols. Figure 19-13 shows the DPLL receiver block; Figure 19-14 shows the transmitter block diagram.

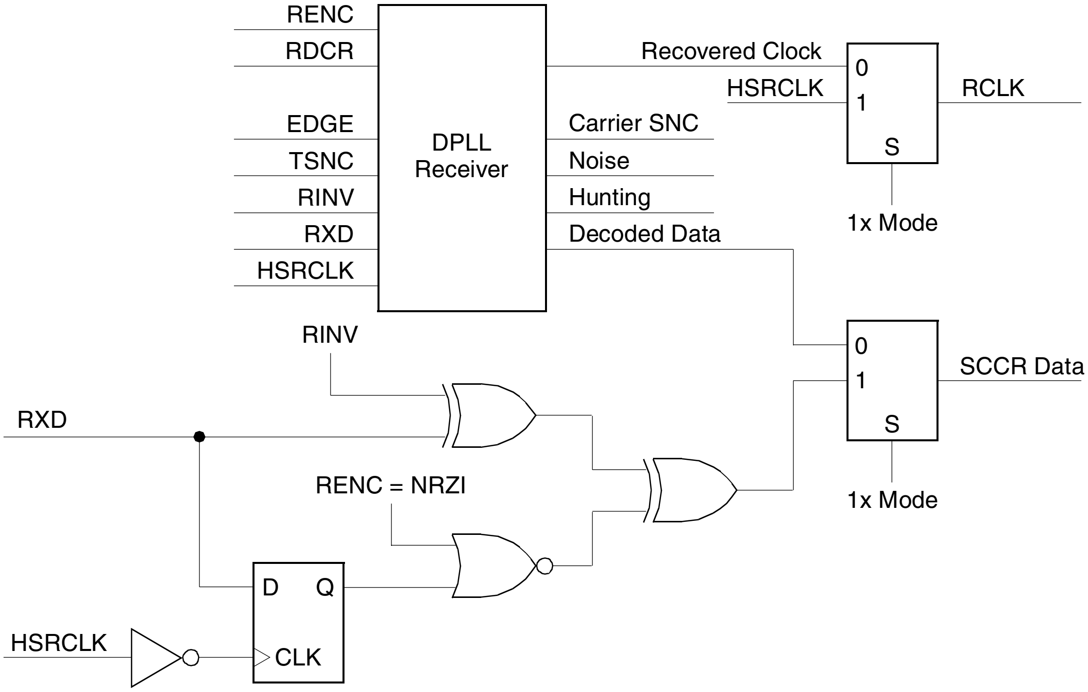


Figure 19-13. DPLL Receiver Block Diagram

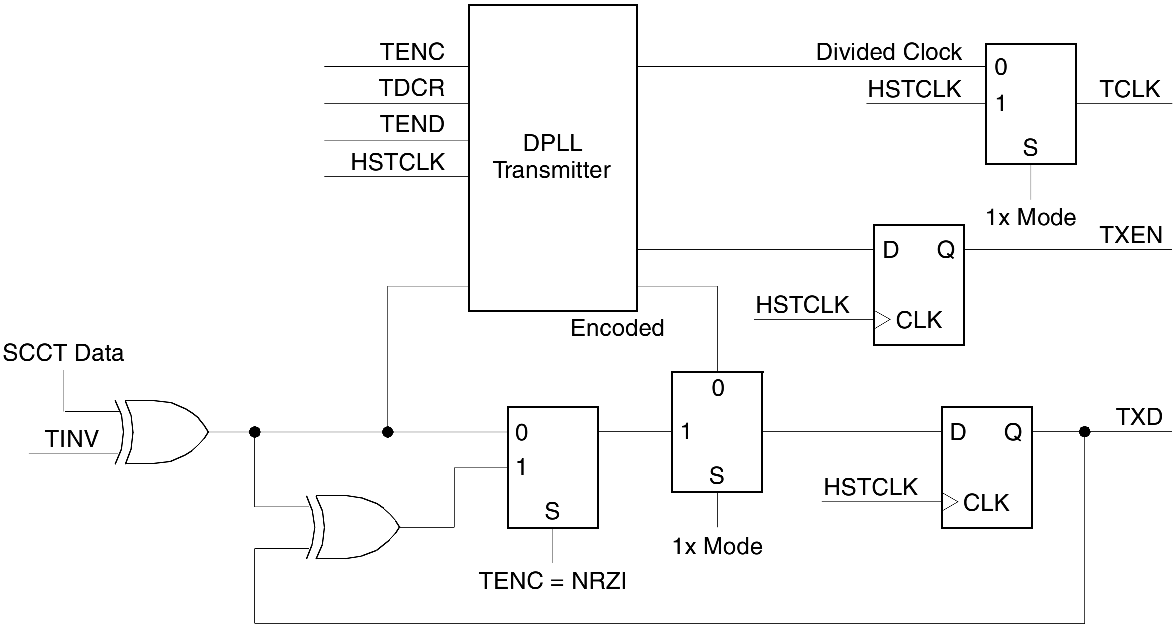


Figure 19-14. DPLL Transmitter Block Diagram

The DPLL can be driven by one of the baud rate generator outputs or an external clock, CLK*x*. In the block diagrams, this clock is labeled HSRCLK/HSTCLK. The HSRCLK/HSTCLK should be approximately 8x, 16x, or 32x the data rate, depending on the coding chosen. The DPLL uses this clock, along with the data stream, to construct a data clock that can be used as the SCC Rx and/or Tx clock. In all modes, the DPLL uses the input clock to determine the nominal bit time. If the DPLL is bypassed, HSRCLK/HSTCLK is used directly as RCLK/TCLK.

At the beginning of operation, the DPLL is in search mode, whereas the first transition resets the internal DPLL counter and begins DPLL operation. While the counter is counting, the DPLL watches the incoming data stream for transitions; when one is detected, the DPLL adjusts the count to produce an output clock that tracks incoming bits.

The DPLL has a carrier-sense signal that indicates when data transfers are on RXD. The carrier-sense signal asserts as soon as a transition is detected on RXD; it negates after the programmed number of clocks in GSMR_L[TSNC] when no transitions are detected.

To prevent itself from locking on the wrong edges and to provide fast synchronization, the DPLL should receive a preamble pattern before it receives the data. In some protocols, the preceding flags or syncs can function as a preamble; others use the patterns in Table 19-8. When transmission occurs, the SCC can generate preamble patterns, as programmed in GSMR_L[TPP, TPL].

Table 19-8. Preamble Requirements

Decoding Method	Preamble Pattern	Minimum Preamble Length Required
NRZI mark	All zeros	8-bit
NRZI space	All ones	8-bit
FM0	All ones	8-bit

Table 19-8. Preamble Requirements (continued)

Decoding Method	Preamble Pattern	Minimum Preamble Length Required
FM1	All zeros	8-bit
Manchester	101010...10	8-bit
Differential Manchester	All ones	8-bit

The DPLL can also be used to invert the data stream of a transfer. This feature is available in all encodings, including standard NRZ format. Also, when the transmitter is idling, the DPLL can either force TXD high or continue encoding the data supplied to it.

The DPLL is used for UART encoding/decoding, which gives the option of selecting the divide ratio in the UART decoding process (8×, 16×, or 32×). Typically, 16× is used.

Note the 1:4 system clock/serial clock ratio does not apply when the DPLL is used to recover the clock in the 8×, 16×, or 32× modes. Synchronization occurs internally after the DPLL generates the Rx clock. Therefore, even the fastest DPLL clock generation (the 8× option) easily meets the required 1:4 ratio clocking limit.

19.3.6.1 Encoding Data with a DPLL

Each SCC contains a DPLL unit that can be programmed to encode and decode the SCC data as NRZ, NRZI Mark, NRZI Space, FM0, FM1, Manchester, and differential Manchester. Figure 19-15 shows the different encoding methods.

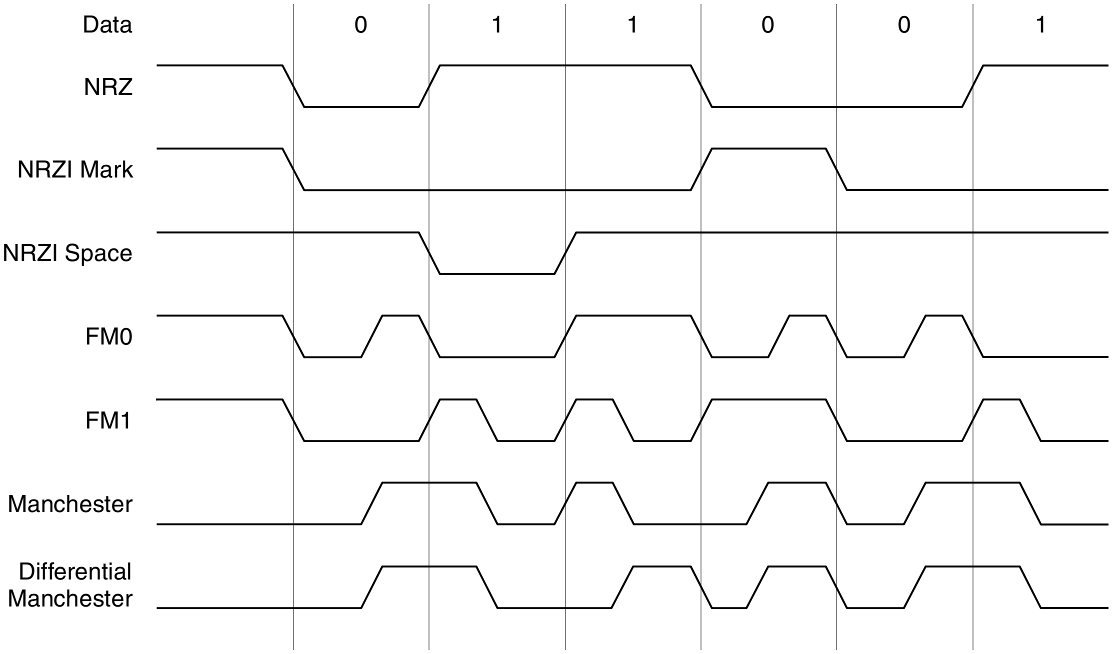


Figure 19-15. DPLL Encoding Examples

If the DPLL is not needed, NRZ or NRZI codings can be selected in GSMR_L[RENC, TENC]. Coding definitions are shown in Table 19-9.

Table 19-9. DPLL Codings

Coding	Description
NRZ	A one is represented by a high level for the duration of the bit and a zero is represented by a low level.
NRZI Mark	A one is represented by no transition at all. A zero is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed).
NRZI Space	A one is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed). A zero is represented by no transition at all.
FM0	A one is represented by a transition only at the beginning of the bit. A zero is represented by a transition at the beginning of the bit and another transition at the center of the bit.
FM1	A one is represented by a transition at the beginning of the bit and another transition at the center of the bit. A zero is represented by a transition only at the beginning of the bit.
Manchester	A one is represented by a high-to-low transition at the center of the bit. A zero is represented by a low to high transition at the center of the bit. In both cases there may be a transition at the beginning of the bit to set up the level required to make the correct center transition.
Differential Manchester	A one is represented by a transition at the center of the bit with the opposite direction from the transition at the center of the preceding bit. A zero is represented by a transition at the center of the bit with the same polarity from the transition at the center of the preceding bit.

19.3.7 Reconfiguring the SCCs

The proper reconfiguration sequence must be followed for SCC parameters that cannot be changed dynamically. For instance, the internal baud rate generators allow on-the-fly changes, but the DPLL-related GSMR does not. The steps in the following sections show how to disable, reconfigure and re-enable an SCC to ensure that buffers currently in use are properly closed before reconfiguring the SCC and that subsequent data goes to or from new buffers according to the new configuration.

Modifying parameter RAM does not require the SCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

19.3.7.1 General Reconfiguration Sequence for an SCC Transmitter

An SCC transmitter can be reconfigured by following the general steps below:

1. If the SCC is sending data, issue a STOP TRANSMIT command. Transmission should stop smoothly. If the SCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed) or the INIT TX PARAMETERS command is issued, the STOP TRANSMIT command is not required.
2. Clear GSMR_L[ENT] to disable the SCC transmitter and put it in reset state.
3. Modify SCC Tx parameters or parameter RAM. To switch protocols or restore the initial Tx parameters, issue an INIT TX PARAMETERS command.

4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.
5. Set GSMR_L[ENT]. Transmission begins using the TxBD pointed to by TBPTR, assuming the R bit is set.

19.3.7.2 Reset Sequence for an SCC Transmitter

The following steps reinitialize an SCC transmit parameters to the reset state:

1. Clear GSMR_L[ENT].
2. Make any modifications then issue the INIT TX PARAMETERS command.
3. Set GSMR_L[ENT].

19.3.7.3 General Reconfiguration Sequence for an SCC Receiver

An SCC receiver can be reconfigured by following these steps:

1. Clear GSMR_L[ENR]. The SCC receiver is now disabled and put in a reset state.
2. Modify SCC Rx parameters or parameter RAM. To switch protocols or restore Rx parameters to their initial state, issue an INIT RX PARAMETERS command.
3. If the INIT RX PARAMETERS command was not issued in step 2, issue an ENTER HUNT MODE command.
4. Set GSMR_L[ENR]. Reception begins using the RxBD pointed to by RBPTR, assuming the E bit is set.

19.3.7.4 Reset Sequence for an SCC Receiver

To reinitialize the SCC receiver to the state it was in after reset, follow the steps below:

1. Clear GSMR_L[ENR].
2. Make any modifications then issue the INIT RX PARAMETERS command.
3. Set GSMR_L[ENR].

19.3.7.5 Switching Protocols

To switch an SCC's protocol without resetting the board or affecting other SCCs, follow the steps below:

1. Clear GSMR_L[ENT, ENR].
2. Make protocol changes in the GSMR and additional parameters, and issue the INIT TX and RX PARAMETERS command to initialize both Tx and Rx parameters.
3. Set GSMR_L[ENT, ENR] to enable the SCC with the new protocol.

19.3.8 Saving Power

To save power when not in use, an SCC can be disabled by clearing GSMR_L[ENT, ENR].

Chapter 20
SCC UART Mode

The universal asynchronous receiver transmitter (UART) protocol is commonly used to send low-speed data between devices. The term asynchronous is used because it is not necessary to send clocking information along with the data being sent. UART links are typically 38400 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 20-1.

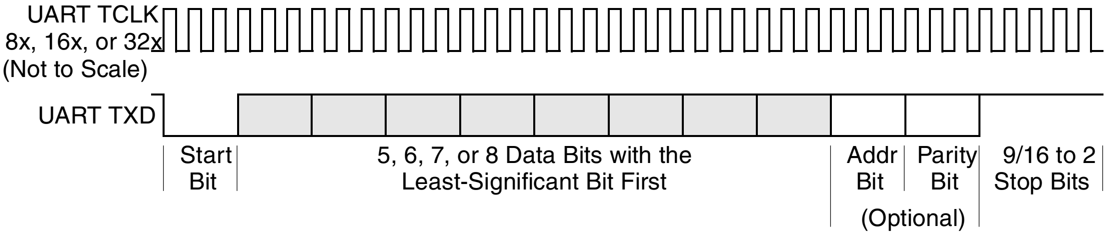


Figure 20-1. UART Character Format

Because the transmitter and receiver operate asynchronously, there is no need to connect the transmit and receive clocks. Instead, the receiver oversamples the incoming data stream (usually by a factor of 16) and uses some of these samples to determine the bit value. Traditionally, the middle 3 of the 16 samples are used. Two UARTs can communicate using this system if the transmitter and receiver use the same parameters, such as the parity scheme and character length.

When data is not sent, a continuous stream of ones is sent (idle condition). Because the start bit is always a zero, the receiver can detect when real data is once again on the line. UART specifies an all-zeros break character, which ends a character transfer sequence.

The most popular protocol that uses asynchronous characters is the RS-232 standard, which specifies baud rates, handshaking protocols, and mechanical/electrical details. Another popular format is RS-485, which defines a balanced line system allowing longer cables than RS-232 links. Even synchronous protocols like HDLC are sometimes defined to run over asynchronous links. The Profibus standard extends UART protocol to include LAN-oriented features such as token passing.

All standards provide handshaking signals, but some systems require only three physical lines—Tx data, Rx data, and ground. Many proprietary standards have been built around the UART's asynchronous character frame, some of which implement a multidrop configuration where multiple stations, each with a specific address, can be present on a network. In multidrop mode, frames of characters are broadcast with the first character acting as a destination address. To accommodate this, the UART frame is extended one bit to distinguish address characters from normal data characters.

In synchronous UART (isochronous operation), a separate clock signal is explicitly provided with the data. Start and stop bits are present in synchronous UART, but oversampling is not required because the clock is provided with each bit.

The general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines. Using standard asynchronous bit rates and protocols, an SCC UART controller can communicate with any existing RS-232-type device and provides a serial communications port to other microprocessors and terminals (either locally or through modems). The independent transmit and receive sections, whose operations are asynchronous with the core, send data from memory (either internal or external) to TXD and receive data from RXD. The UART controller supports a multidrop mode for master/slave operations with 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.

20.1 Features

The following list summarizes main features of an SCC UART controller:

- Flexible message-based data structure
- Implements synchronous and asynchronous UART
- Multidrop operation
- Receiver wake-up on idle line or address bit
- Receive entire messages into buffers as indicated by receiver idle timeout or by control character reception
- Eight control character comparison
- Two address comparison in multidrop configurations
- Maintenance of four 16-bit error counters
- Received break character length indication
- Programmable data length (5–8 bits)
- Programmable fractional stop bit lengths (9/16–2 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

20.2 Normal Asynchronous Mode

In normal asynchronous mode, the receive shift register receives incoming data on RXDx. Control bits in the UART mode register (PSMR) define the length and format of the UART character. Bits are received in the following order:

1. Start bit
2. 5–8 data bits (lsb first)

3. Address/data bit (optional)
4. Parity bit (optional)
5. Stop bits

The receiver uses a clock 8×, 16×, or 32× faster than the baud rate and samples each bit of the incoming data three times around its center. The value of the bit is determined by the majority of those samples; if all do not agree, the noise indication counter (NOSEC) in parameter RAM is incremented. When a complete character has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields.

The SCC can receive fractional stop bits. The next character's start bit can begin any time after the three middle samples are taken. The UART transmit shift register sends outgoing data on TXDx. Data is then clocked synchronously with the transmit clock, which may have either an internal or external source. Characters are sent lsb first. Only the data portion of the UART frame is stored in the buffers because start and stop bits are generated and stripped by the SCC. A parity bit can be generated in transmission and checked during reception; although it is not stored in the buffer, its value can be inferred from the buffer's reporting mechanism. Similarly, the optional address bit is not stored in the transmit or receive buffer, but is supplied in the BD itself. Parity generation and checking includes the optional address bit. GSMR_H[RFW] must be set for an 8-bit receive FIFO in the UART receiver.

20.3 Synchronous Mode

In synchronous mode, the controller uses a 1× data clock for timing. The receive shift register receives incoming data on RXDx synchronous with the clock. The bit length and format of the serial character are defined by the control bits in the PSMR in the same way as in asynchronous mode. When a complete byte has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields. GSMR_H[RFW] must be set for an 8-bit receive FIFO.

The synchronous UART transmit shift register sends outgoing data on TXDx. Data is then clocked synchronously with the transmit clock, which can have an internal or external source.

20.4 SCC UART Parameter RAM

For UART mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 20-1.

Table 20-1. UART-Specific SCC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Dword	Reserved
0x38	**MAX_IDL**	Hword	Maximum idle characters. When a character is received, the receiver begins counting idle characters. If MAX_IDL idle characters are received before the next data character, an idle timeout occurs and the buffer is closed, generating a maskable interrupt request to the core to receive the data from the buffer. Thus, MAX_IDL offers a way to demarcate frames. To disable the feature, clear MAX_IDL. The bit length of an idle character is calculated as follows: 1 + data length (5–9) + 1 (if parity is used) + number of stop bits (1–2). For 8 data bits, no parity, and 1 stop bit, the character length is 10 bits.
0x3A	IDLC	Hword	Temporary idle counter. Holds the current idle count for the idle timeout process. IDLC is a down-counter and does not need to be initialized or accessed.
0x3C	**BRKCR**	Hword	Break count register (transmit). Determines the number of break characters the transmitter sends. The transmitter sends a break character sequence when a STOP TRANSMIT command is issued. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character consists of 10 zero bits.
0x3E	**PAREC**	Hword	User-initialized,16-bit (modulo–2^{16}) counters incremented by the CP. PAREC counts received parity errors. FRMEC counts received characters with framing errors. NOSEC counts received characters with noise errors. BRKEC counts break conditions on the signal. A break condition can last for hundreds of bit times, yet BRKEC is incremented only once during that period.
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	Temp 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 FIFO without affecting a Tx buffer in progress. See Section 20.11, "Inserting Control Characters into the Transmit Data Stream."
0x50	**CHARACTER1**	Hword	Control character 1–8. These characters define the Rx control characters on which interrupts can be generated.
0x52	**CHARACTER2**	Hword	
0x54	**CHARACTER3**	Hword	
0x56	**CHARACTER4**	Hword	
0x58	**CHARACTER5**	Hword	
0x5A	**CHARACTER6**	Hword	
0x5C	**CHARACTER7**	Hword	
0x5E	**CHARACTER8**	Hword	

Table 20-1. UART-Specific SCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x4C	RTEMP	Hword	Temp 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 PSMR[RZS] = 1; holds the last break character pattern. By counting zeros in RLBC, the core can measure break length to a one-bit resolution. Read RLBC by counting the zeros written from bit 0 to where the first one was written. RLBC = 0b001xxxxxxxxxxxxx indicates two zeros; 0b1xxxxxxxxxxxxxxx indicates no zeros. Note that RLBC can be used in combination with BRKLN above to calculate the number of bits in the break sequence: (BRKLN * character length) + (number of zeros in RLBC).

[1] From SCC base. See Section 19.3.1, “SCC Base Addresses.”

20.5 Data-Handling Methods: Character- or Message-Based

An SCC UART controller uses the same BD table and buffer structures as the other protocols and supports both multibuffer, message-based and single-buffer, character-based operation.

For character-based transfers, each character is sent with stop bits and parity and received into separate 1-byte buffers. A maskable interrupt is generated when each buffer is received.

In a message-based environment, transfers can be made on entire messages rather than on individual characters. To simplify programming and save processor overhead, a message is transferred as a linked list of buffers without core intervention. For example, before handling input data, a terminal driver may wait for an end-of-line character or an idle timeout rather than be interrupted when each character is received. Conversely, ASCII files can be sent as messages ending with an end-of-line character.

When receiving messages, up to eight control characters can be configured to mark the end of a message or generate a maskable interrupt without being stored in the buffer. This option is useful when flow control characters such as XON or XOFF are needed but are not part of the received message. See Section 20.9, “Receiving Control Characters.”

20.6 Error and Status Reporting

Overrun, parity, noise, and framing errors are reported through the BDs and/or error counters in the UART parameter RAM. Signal status is indicated in the status register; a maskable interrupt is generated when status changes.

20.7 SCC UART Commands

The transmit commands in Table 20-2 are issued to the CP command register (CPCR).

Table 20-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 8 Tx clocks. STOP TRANSMIT disables character transmission. If the SCC receives STOP TRANSMIT as a message is being sent, the message is aborted. The transmitter finishes sending data transferred to its FIFO and stops. The TBPTR is not advanced. The UART transmitter sends a programmable break sequence and starts sending idles. The number of break characters in the sequence (which can be zero) should be written to BRKCR in the parameter RAM before issuing this command.
GRACEFUL STOP TRANSMIT	Used to stop transmitting smoothly. The transmitter stops after the current buffer has been completely sent or immediately if no buffer is being sent. SCCE[GRA] is set once transmission stops, then the UART Tx parameters, including the TxBD, can be modified. TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables transmission. The controller expects this command after it disables the channel in its PSMR, after a STOP TRANSMIT command, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. Transmission resumes from the current BD.
INIT TX PARAMETERS	Resets the transmit parameters in the parameter RAM. Issue only when the transmitter is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 20-3.

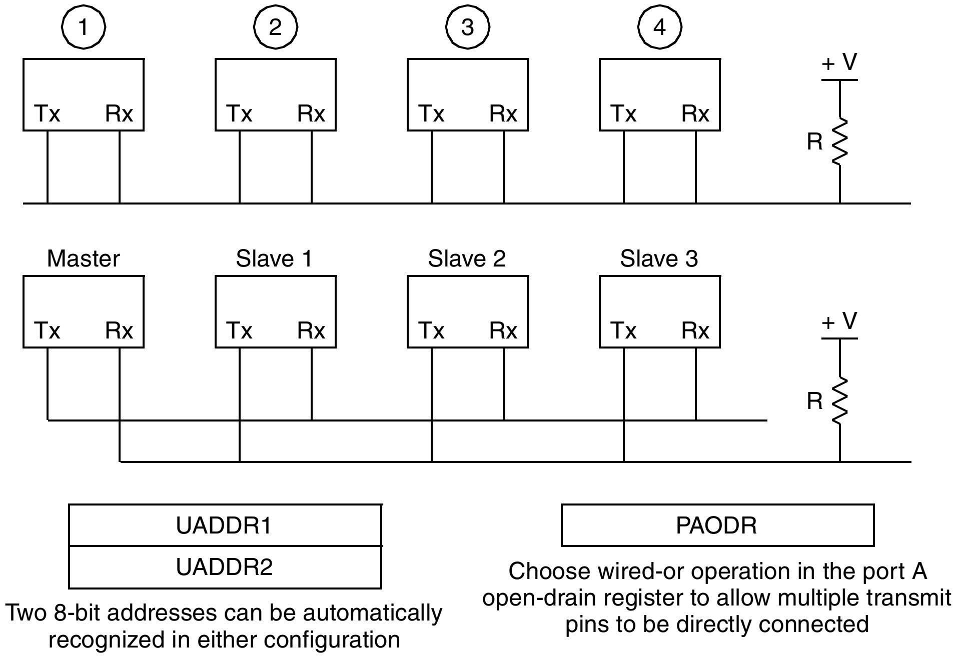
Table 20-3. Receive Commands

Command	Description
ENTER HUNT MODE	Forces the receiver to close the RxBD in use and enter hunt mode. After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. If a message is in progress, the receiver continues receiving in the next BD. In multidrop hunt mode, the receiver continually scans the input data stream for the address character. When it is not in multidrop mode, it waits for the idle sequence (one character of idle). Data present in the Rx FIFO is not lost when this command is executed.
CLOSE RXBD	Forces the SCC to close the RxBD in use and use the next BD for subsequent received data. If the SCC is not in the process of receiving data, no action is taken. Note that in an SCC UART controller, CLOSE RXBD functions like ENTER HUNT MODE but does not need to receive an idle character to continue receiving.
INIT RX PARAMETERS	Resets the receive parameters in the parameter RAM. Should be issued when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

20.8 Multidrop Systems and Address Recognition

In multidrop systems, more than two stations can be on a network, each with a specific address. Figure 20-2 shows two examples of this configuration. Frames made up of many characters can be broadcast as long as the first character is the destination address. The UART frame is extended by one bit to distinguish an address character from standard data characters. Programmed in PSMR[UM], the controller supports the following two multidrop modes:

- Automatic multidrop mode—The controller checks the incoming address character and accepts subsequent data only if the address matches one of two user-defined values. The two 16-bit address registers, UADDR1 and UADDR2, support address recognition. Only the lower 8 bits are used so the upper 8 bits should be cleared; for addresses less than 8 bits, unused high-order bits should also be cleared. The incoming address is checked against UADDR1 and UADDR2. When a match occurs, RxBD[AM] indicates whether UADDR1 or UADDR2 matched.
- Manual multidrop mode—The controller receives all characters. An address character is always written to a new buffer and can be followed by data characters. User software performs the address comparison.



Two 8-bit addresses can be automatically recognized in either configuration

Choose wired-or operation in the port A open-drain register to allow multiple transmit pins to be directly connected

Figure 20-2. Two UART Multidrop Configurations

20.9 Receiving Control Characters

The UART receiver can recognize special control characters used in a message-based environment. Eight control characters can be defined in a control character table in the UART parameter RAM. Each incoming character is compared to the table entries using a mask (the received control character mask, RCCM) to strip don’t cares. If a match occurs, the received control character can either be written to the receive buffer or rejected.

If the received control character is not rejected, it is written to the receive buffer. The receive buffer is then automatically closed to allow software to handle end-of-message characters. Control characters that are not part of the actual message, such as XOFF, can be rejected. Rejected characters bypass the receive buffer and are written directly to the received control character register (RCCR), which triggers maskable interrupt.

The 16-bit entries in the control character table support control character recognition. Each entry consists of the control character, a valid bit (end of table), and a reject bit. See Figure 20-3.

Offset [1]	0	1	2 7	8 15
0x50	**E**	**R**	—	**CHARACTER1**
0x52	**E**	**R**	—	**CHARACTER2**
• • •	• • •	• • •	• • •	• • •
0x5E	**E**	**R**	—	**CHARACTER8**
0x60	**1**	**1**	—	**RCCM**
0x62	—			RCCR

[1] From SCC*x* base address.

Figure 20-3. Control Character Table

Table 20-4 describes the data structure used in control character recognition.

Table 20-4. Control Character Table, RCCM, and RCCR Descriptions

Offset	Bits	Name	Description
0x50– 0x5E	0	**E**	End of table. In tables with eight control characters, E is always 0. 0 This entry is valid. 1 The entry is not valid and is not used.
	1	**R**	Reject character. 0 A matching character is not rejected but is written into the Rx buffer, which is then closed. If RxBD[I] is set, the buffer closing generates a maskable interrupt through SCCE[RX]. A new buffer is opened if more data is in the message. 1 A matching character is written to RCCR and not to the Rx buffer. A maskable interrupt is generated through SCCE[CCR]. The current Rx buffer is not closed.
	2–7	—	Reserved
	8–15	**CHARACTERn**	Control character values 1–8. Defines control characters to be compared to the incoming character. For characters smaller than 8 bits, the most significant bits should be zero.
0x60	0–1	**0b11**	Must be set. Used to mark the end of the control character table in case eight characters are used. Setting these bits ensures correct operation during control character recognition.
	2–7	—	Reserved
	8–15	**RCCM**	Received control character mask. Used to mask the comparison of CHARACTER*n*. Each RCCM bit corresponds to the respective bit of CHARACTER*n* and decodes as follows. 0 Ignore this bit when comparing the incoming character to CHARACTER*n*. 1 Use this bit when comparing the incoming character to CHARACTER*n*.
0x62	0–7	—	Reserved
	8–15	RCCR	Received control character register. If the newly arrived character matches and is rejected from the buffer (R = 1), the PIP controller writes the character into the RCCR and generates a maskable interrupt. If the core does not process the interrupt and read RCCR before a new control character arrives, the previous control character is overwritten.

20.10 Hunt Mode (Receiver)

A UART receiver in hunt mode remains deactivated until an idle or address character is recognized, depending on PSMR[UM]. A receiver is forced into hunt mode by issuing an ENTER HUNT MODE command.

The receiver aborts any message in progress when ENTER HUNT MODE is issued. When the message is finished, the receiver is reenabled by detecting the idle line (one idle character) or by the address bit of the next message, depending on PSMR[UM]. When a receiver in hunt mode receives a break sequence, it increments BRKEC and generates a BRK interrupt condition.

20.11 Inserting Control Characters into the Transmit Data Stream

The SCC UART transmitter can send out-of-sequence, flow-control characters like XON and XOFF. The controller polls the transmit out-of-sequence register (TOSEQ), shown in Figure 20-4, whenever the transmitter is enabled for UART operation, including during a UART freeze operation, UART buffer transmission, and when no buffer is ready for transmission. The TOSEQ character (in CHARSEND) is sent at a higher priority than the other characters in the transmit buffer, but does not preempt characters already in the transmit FIFO. This means that the XON or XOFF character may not be sent for eight or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—		**REA**	**I**	CT	—		**A**	**CHARSEND**							
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	SCC base + 0x4E															

Figure 20-4. Transmit Out-of-Sequence Register (TOSEQ)

Table 20-5 describes TOSEQ fields.

Table 20-5. 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 CP clears this bit after transmission.
3	**I**	Interrupt. If this bit is set, transmission completion is flagged in the event register (SCCE[TX] is set), triggering a maskable interrupt to the core.
4	CT	Clear-to-send lost. Operates only if the SCC monitors $\overline{\text{CTS}}$ (GSMR_L[DIAG]). The CP sets this bit if $\overline{\text{CTS}}$ negates when the TOSEQ character is sent. If $\overline{\text{CTS}}$ negates and the TOSEQ character is sent during a buffer transmission, the TxBD[CT] status bit is also set.
5–6	—	Reserved, should be cleared.e

Table 20-5. 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.

20.12 Sending a Break (Transmitter)

A break is an all-zeros character with no stop bit that is sent by issuing a STOP TRANSMIT command. The SCC finishes transmitting outstanding data, sends a programmable number of break characters (determined by BRKCR), and reverts to idle or sends data if a RESTART TRANSMIT command is given before completion. When the break code is complete, the transmitter sends at least one high bit before sending more data, to guarantee recognition of a valid start bit. Because break characters do not preempt characters in the transmit FIFO, they may not be sent for eight (SCC) or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

20.13 Sending a Preamble (Transmitter)

Sending a preamble sequence of consecutive ones ensures that a line is idle before sending a message. If the preamble bit TxBD[P] is set, the SCC sends a preamble sequence (idle character) before sending the buffer. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones is sent before the first character in the buffer.

20.14 Fractional Stop Bits (Transmitter)

The asynchronous UART transmitter, shown in Figure 20-5, can be programmed to send fractional stop bits. The FSB field in the data synchronization register (DSR) determines the fractional length of the last stop bit to be sent. FSB can be modified at any time. If two stop bits are sent, only the second is affected. Idle characters are always sent as full-length characters

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—	**FSB**				—	—	—	—	—	—	—	—	—	—	—
Reset	0	1111				1	1	0	0	1	1	1	1	1	1	0
R/W	R/W															
Addr																

Figure 20-5. Asynchronous UART Transmitter

Table 20-6 describes DSR fields.

Table 20-6. DSR Fields Descriptions

Bit	Name	Description
0	—	0b0
1–4	**FSB**	Fractional stop bits. For 16× oversampling: 1111 Last transmitted stop bit 16/16. Default value after reset. 1110 Last transmitted stop bit 15/16 … 1000 Last transmitted stop bit 9/16 0xxx Invalid. Do not use. For 32× oversampling: 1111 Last transmitted stop bit 32/32. Default value after reset. 1110 Last transmitted stop bit 31/32 … 0000 Last transmitted stop bit 17/32 For 8× oversampling: 1111 Last transmitted stop bit 8/8. Default value after reset. 1110 Last transmitted stop bit 7/8 1101 Last transmitted stop bit 6/8 1100 Last transmitted stop bit 5/8 10xx Invalid. Do not use. 0xxx Invalid. Do not use. The UART receiver can always receive fractional stop bits. The next character's start bit can begin any time after the three middle samples have been taken.
5–6	—	0b11
7–8	—	0b00
9–14	—	0b111111
15	—	0b0

20.15 Handling Errors in the SCC UART Controller

The UART controller reports character reception and transmission error conditions through the BDs, the error counters, and the SCCE. Modem interface lines can be monitored by the port C pins. Transmission errors are described in Table 20-7.

Table 20-7. Transmission Errors

Error	Description
$\overline{\text{CTS}}$ lost during character transmission	When $\overline{\text{CTS}}$ negates during transmission, the channel stops after finishing the current character. The CP sets TxBD[CT] and generates the TX interrupt if it is not masked. The channel resumes transmission after the RESTART TRANSMIT command is issued and $\overline{\text{CTS}}$ is asserted. Note that if $\overline{\text{CTS}}$ is used, the UART also offers an asynchronous flow control option that does not generate an error. See the description of PSMR[FLC] in Table 20-9.

Reception errors are described in Table 20-8.

Table 20-8. Reception Errors

Error	Description
Overrun	Occurs when the channel overwrites the previous character in the Rx FIFO with a new character, losing the previous character. The channel then writes the new character to the buffer, closes it, sets RxBD[OV], and generates an RX interrupt if not masked. In automatic multidrop mode, the receiver enters hunt mode immediately.
$\overline{\text{CD}}$ lost during character reception	If this error occurs and the channel is using this pin to automatically control reception, the channel terminates character reception, closes the buffer, sets RxBD[CD], and generates the RX interrupt if not masked. This error has the highest priority. The last character in the buffer is lost and other errors are not checked. In automatic multidrop mode, the receiver enters the hunt mode immediately.
Parity	When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets RxBD[PR], and generates the RX interrupt if not masked. The channel also increments the parity error counter PAREC. In automatic multidrop mode, the receiver enters hunt mode immediately.
Noise	A noise error occurs when the three samples of a bit are not identical. When this error occurs, the channel writes the received character to the buffer, proceeds normally, but increments the noise error counter NOSEC. Note that this error does not occur in synchronous mode.
Idle sequence receive	If the UART is receiving data and gets an idle character (all ones), the channel begins counting consecutive idle characters received. If MAX_IDL is reached, the buffer is closed and an RX interrupt is generated if not masked. If no buffer is open, this event does not generate an interrupt or any status information. The internal idle counter (IDLC) is reset every time a character is received. To disable the idle sequence function, clear MAX_IDL.
Framing	The UART reports a framing errors when it receives a character with no stop bit, regardless of the mode. The channel writes the received character to the buffer, closes it, sets RxBD[FR], generates the RX interrupt if not masked, increments FRMEC, but does not check parity for this character. In automatic multidrop mode, the receiver immediately enters hunt mode. If the UART allows data with no stop bits (PSMR[RZS] = 1) when in synchronous mode (PSMR[SYN] = 1), framing errors are reported but reception continues assuming the unexpected zero is the start bit of the next character; in this case, the user may ignore a reported framing error until multiple framing errors occur within a short period.
Break sequence	When the first break sequence is received, the UART increments the break error counter BRKEC. It updates BRKLN when the sequence completes. After the first 1 is received, the UART sets SCCE[BRKE], which generates an interrupt if not masked. If the UART is receiving characters when it receives a break, it closes the Rx buffer, sets RxBD[BR], and sets SCCE[RX], which can generate an interrupt if not masked. If PSMR[RZS] = 1 when the UART is in synchronous mode, a break sequence is detected after two successive break characters are received.

20.16 UART Mode Register (PSMR)

For UART mode, the SCC protocol-specific mode register (PSMR) is called the UART mode register. Many bits can be modified while the receiver and transmitter are enabled. Figure 20-6 shows the PSMR in UART mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FLC	SL	CL		UM		FRZ	RZS	SYN	DRT	—	PEN	RPM		TPM	
Reset	0															
R/W	R/W															
Addr	0x11A08 (PSMR1); 0x11A48 (PSMR3); 0x11A68 (PSMR4)															

Figure 20-6. Protocol-Specific Mode Register for UART (PSMR)

Table 20-9 describes PSMR UART fields.

Table 20-9. PSMR UART Field Descriptions

Bit	Name	Description
0	FLC	Flow control. 0 Normal operation. The GSMR and port C registers determine the mode of $\overline{\text{CTS}}$. 1 Asynchronous flow control. When $\overline{\text{CTS}}$ is negated, the transmitter stops at the end of the current character. If $\overline{\text{CTS}}$ is negated past the middle of the current character, the next full character is sent before transmission stops. When $\overline{\text{CTS}}$ is asserted again, transmission continues where it left off and no $\overline{\text{CTS}}$ lost error is reported. Only idle characters are sent while $\overline{\text{CTS}}$ is negated.
1	SL	Stop length. Selects the number of stop bits the SCC sends. SL can be modified on-the-fly. The receiver is always enabled for one stop bit unless the SCC UART is in synchronous mode and PSMR[RZS] is set. Fractional stop bits are configured in the DSR. 0 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 CPM compares the address of an incoming address character with UADDR*x* parameter RAM values; subsequent data is accepted only if a match occurs.
6	FRZ	Freeze transmission. Allows the UART transmitter to pause and later continue from that point. 0 Normal operation. If the buffer was previously frozen, it resumes transmission from the next character in the same buffer that was frozen. 1 The SCC completes transmission of any data already transferred to the Tx FIFO (the number of characters depends on GSMR_H[TFL]) and then freezes. After FRZ is cleared, transmission resumes from the next character.

Table 20-9. PSMR UART Field Descriptions (continued)

Bit	Name	Description
7	RZS	Receive zero stop bits. 0 The receiver operates normally, but at least one stop bit is needed between characters. A framing error is issued if a stop bit is missing. Break status is set if an all-zero character is received with a zero stop bit. 1 Configures the receiver to receive data without stop bits. Useful in V.14 applications where SCC UART controller data is supplied synchronously and all stop bits of a particular character can be omitted for cross-network rate adaptation. RZS should be set only if SYN is set. The receiver continues if a stop bit is missing. If the stop bit is a zero, the next bit is considered the first data bit of the next character. A framing error is issued if a stop bit is missing, but a break status is reported only after two consecutive break characters have no stop bits.
8	SYN	Synchronous mode. 0 Normal asynchronous operation. GSMR_L[TENC,RENC] must select NRZ and GSMR_L[TDCR, RDCR] select either 8×, 16×, or 32×. 16× is recommended for most applications. 1 Synchronous SCC UART controller using 1× clock (isochronous UART operation). GSMR_L[TENC, RENC] must select NRZ and GSMR_L[RDCR, TDCR] select 1× mode. A bit is transferred with each clock and is synchronous to the clock, which can be internal or external.
9	DRT	Disable receiver while transmitting. 0 Normal operation 1 While the SCC is sending data, the internal $\overline{\text{RTS}}$ disables and gates the receiver. Useful for a multidrop configuration in which the user does not want to receive its own transmission. For multidrop UART mode, set the BDs' preamble bit, TxBD[P]. **Note:** If DRT = 1, GSMR_H[CDS] should be cleared unless both of the following are true: the same clock is used for TCLK and RCLK, and 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.

20.17 SCC UART Receive Buffer Descriptor (RxBD)

The CPM uses RxBDs to report on each buffer received. The CPM closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- A user-defined control character is received.
- An error occurs during message processing.
- A full receive buffer is detected.
- A MAX_IDL number of consecutive idle characters is received.

- An ENTER HUNT MODE or CLOSE RXBD command is issued.
- An address character is received in multidrop mode. The address character is written to the next buffer for a software comparison.

Figure 20-7 shows an example of how RxBDs are used in receiving.

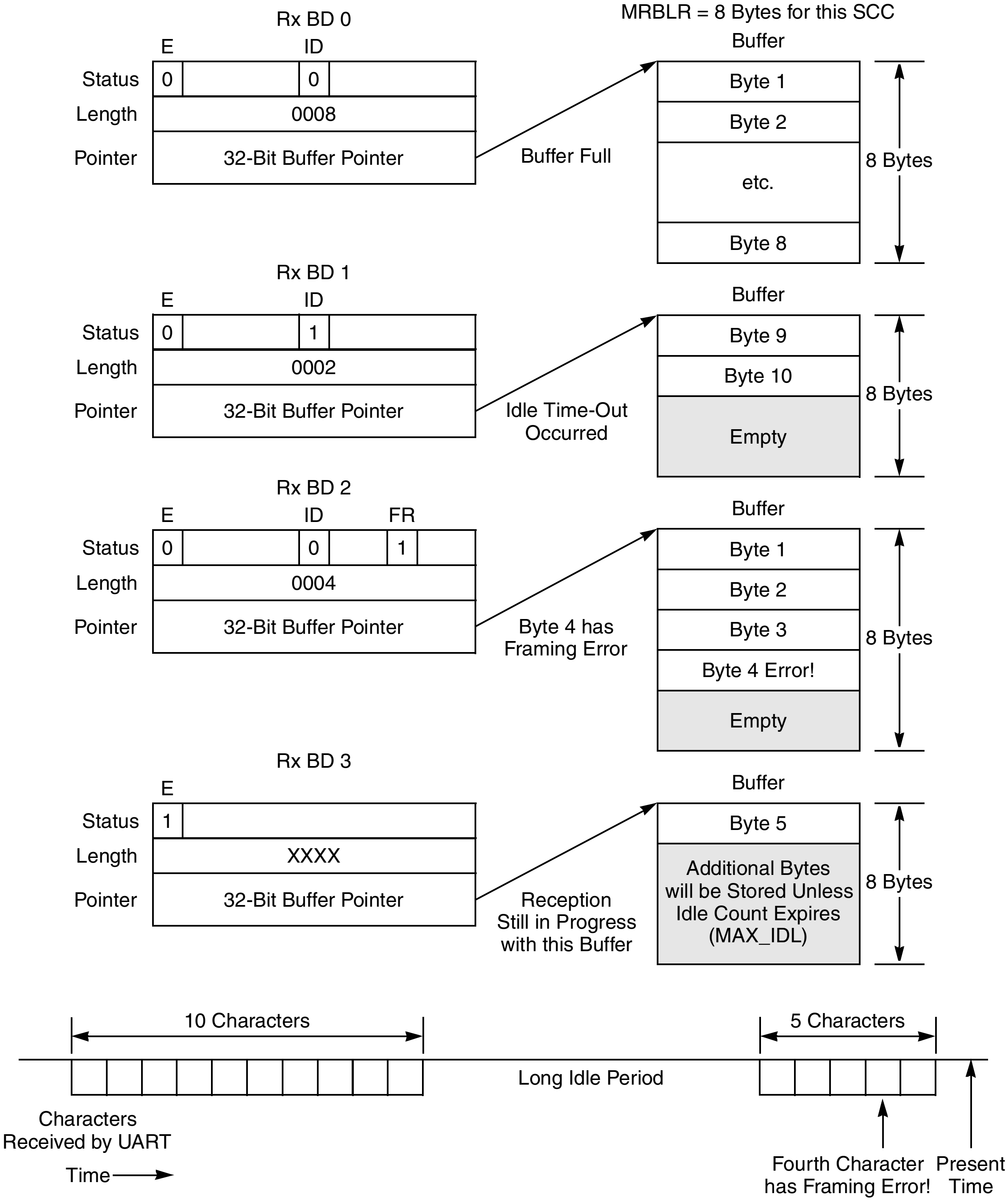


Figure 20-7. SCC UART Receiving using RxBDs

Figure 20-8 shows the SCC UART RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	C	A	**CM**	ID	AM	—	BR	FR	PR	—	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 20-8. SCC UART Receive Buffer Descriptor (RxBD)

Table 20-10 describes RxBD status and control fields.

Table 20-10. SCC UART RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or reception was aborted due to an error. The core can read or write to any fields of this BD. The CPM does not reuse this BD while E = 0. 1 The buffer is not full. The CPM controls this BD and buffer. The core should not modify this BD.
1	—	Reserved, should be cleared
2	**W**	Wrap (last buffer descriptor in the BD table). 0 Not the last descriptor in the table 1 Last descriptor in the table. After this buffer is used, the CPM receives incoming data using the BD pointed to by RBASE. The number of BDs in this table is programmable and determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 The CP sets SCCE[RX] when this buffer is completely filled by the CPM, indicating the need for the core to process the buffer. Setting SCCE[RX] causes an interrupt if not masked.
4	C	Control character. 0 This buffer does not contain a control character. 1 The last byte in this buffer matches a user-defined control character.
5	A	Address. 0 The buffer contains only data. 1 For manual multidrop mode, A indicates the first byte of this buffer is an address byte. Software should perform address comparison. In automatic multidrop mode, A indicates the buffer contains a message received immediately after an address matched UADDR1 or UADDR2. The address itself is not written to the buffer but is indicated by the AM bit.
6	**CM**	Continuous mode. 0 Normal operation. The CPM clears E after this BD is closed. 1 The CPM does not clear E after this BD is closed, allowing the buffer to be overwritten when the CPM accesses this BD again. E is cleared if an error occurs during reception, regardless of CM.
7	ID	Buffer closed on reception of idles. The buffer is closed because a programmable number of consecutive idle sequences (MAX_IDL) was received.
8	AM	Address match. Significant only if the address bit is set and automatic multidrop mode is selected in PSMR[UM]. After an address match, AM identifies which user-defined address character was matched. 0 The address matched the value in UADDR2. 1 The address matched the value in UADDR1.
9	—	Reserved, should be cleared

Table 20-10. SCC UART RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
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 19.2, “SCC Buffer Descriptors (BDs),” describes the data length and buffer pointer fields.

20.18 SCC UART Transmit Buffer Descriptor (TxBD)

The CPM uses BDs to confirm transmission and indicate error conditions so the core knows that buffers have been serviced. Figure 20-9 shows the SCC UART TxBD.

	0	1	2	3	4	5	6	7	8	9	15
Offset + 0	R	—	W	I	CR	A	CM	P	NS	—	CT
Offset + 2	Data Length										
Offset + 4	Tx Buffer Pointer										
Offset + 6											

Figure 20-9. SCC UART Transmit Buffer Descriptor (TxBD)

Table 20-11 describes TxBD status and control fields.

Table 20-11. SCC UART TxBD Status and Control Field Descriptions

Bit	Name	Description
0	**R**	Ready. 0 The buffer is not ready. This BD and buffer can be modified. The CPM automatically clears R after the buffer is sent or an error occurs. 1 The user-prepared buffer is waiting to begin transmission or is being transmitted. Do not modify the BD once R is set.
1	—	Reserved, should be cleared
2	**W**	Wrap (last buffer descriptor in TxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is processed. 1 SCCE[TX] is set after this buffer is processed by the CPM, which can cause an interrupt.

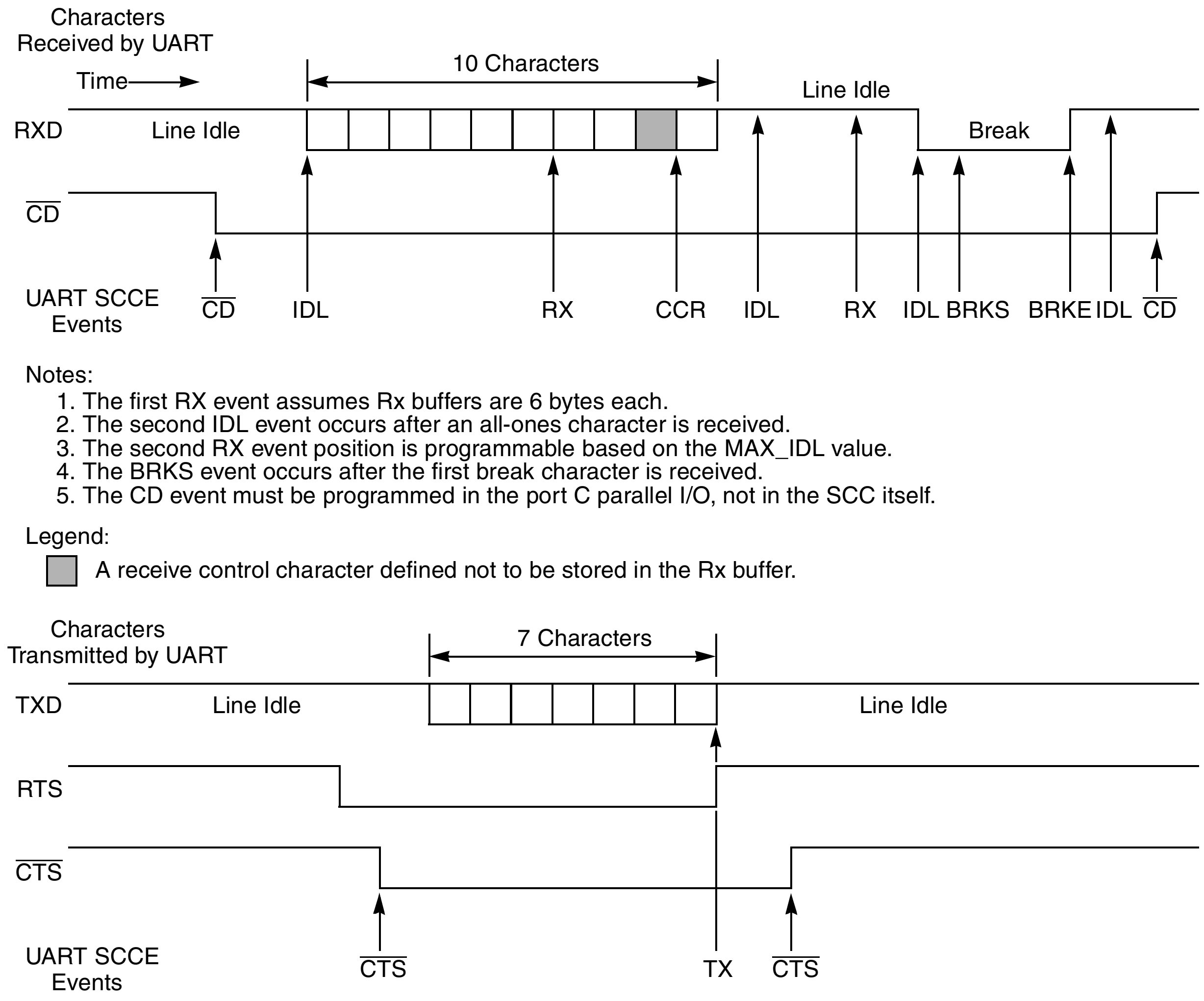
Table 20-11. SCC UART TxBD Status and Control Field Descriptions (continued)

Bit	Name	Description
4	**CR**	Clear-to-send report. 0 The next buffer is sent with no delay (assuming it is ready), but if a $\overline{\text{CTS}}$ lost condition occurs, TxBD[CT] may not be set in the correct TxBD or may not be set at all. Asynchronous flow control, however, continues to function normally. 1 Normal $\overline{\text{CTS}}$ lost error reporting and three bits of idle are sent between consecutive buffers.
5	**A**	Address. Valid only in multidrop mode—automatic or manual. 0 This buffer contains only data. 1 This buffer contains address characters. All data in this buffer is sent as address characters.
6	**CM**	Continuous mode. 0 Normal operation. The CPM clears R after this BD is closed. 1 The CPM does not clear R after this BD is closed, allowing the buffer to be resent next time the CPM accesses this BD. However, R is cleared by transmission errors, regardless of CM.
7	**P**	Preamble. 0 No preamble sequence is sent. 1 Before sending data, the controller sends an idle character consisting of all ones. If the data length of this BD is zero, only a preamble is sent.
8	**NS**	No stop bit or shaved stop bit sent. 0 Normal operation. Stop bits are sent with all characters in this buffer. 1 If PSMR[SYN] = 1, data in this buffer is sent without stop bits. If SYN = 0, the stop bit is shaved, depending on the DSR setting; see Section 20.14, “Fractional Stop Bits (Transmitter).”
9–14	—	Reserved, should be cleared
15	CT	$\overline{\text{CTS}}$ lost. The CPM writes this status bit after sending the associated buffer. 0 $\overline{\text{CTS}}$ remained asserted during transmission 1 $\overline{\text{CTS}}$ negated during transmission

The data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).”

20.19 SCC UART Event Register (SCCE) and Mask Register (SCCM)

The SCC event register (SCCE) is used to report events recognized by the UART channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts can be masked in the UART mask register (SCCM), which has the same format as SCCE. Setting a mask bit enables the corresponding SCCE interrupt; clearing a bit masks it. Figure 20-10 shows example interrupts that can be generated by the SCC UART controller.



Notes:
1. The first RX event assumes Rx buffers are 6 bytes each.
2. The second IDL event occurs after an all-ones character is received.
3. The second RX event position is programmable based on the MAX_IDL value.
4. The BRKS event occurs after the first break character is received.
5. The CD event must be programmed in the port C parallel I/O, not in the SCC itself.

Legend:

▨ A receive control character defined not to be stored in the Rx buffer.

Notes:
1. TX event assumes all seven characters were put into a single buffer and TxBD[CR]=1.
2. The CTS event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 20-10. SCC UART Interrupt Event Example

SCCE bits are cleared by writing ones; writing zeros has no effect. Unmasked bits must be cleared before the CPM clears an internal interrupt request. Figure 20-11 shows SCCE/SCCM for UART operation.

	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—					AB	IDL	GRA	BRKE	BRKS	—	CCR	BSY	TX	RX
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x11A10 (SCCE1); 0x11A50 (SCCE3); 0x11A70 (SCCE4) 0x11A14 (SCCM1); 0x11A54 (SCCM3); 0x11A74 (SCCM4)														

Figure 20-11. SCC UART Event Register (SCCE) and Mask Register (SCCM)

Table 20-12 describes SCCE fields for UART mode.

Table 20-12. SCCE/SCCM Field Descriptions for UART Mode [1]

Bit	Name	Description
0–5	—	Reserved, should be cleared. Refer to note 1 below.
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 16, “Baud-Rate Generators (BRGs).”
7	IDL	Idle sequence status changed. Set when the channel detects a change in the serial line. The line’s real-time status can be read in SCCS[ID]. Idle is entered when a character of all ones is received; it is exited when a zero is received.
8	GRA	Graceful stop complete. Set as soon as the transmitter finishes any buffer in progress after a GRACEFUL STOP TRANSMIT command is issued. It is set immediately if no buffer is in progress.
9	BRKE	Break end. Set when an idle bit is received after a break sequence.
10	BRKS	Break start. Set when the first character of a break sequence is received. Multiple BRKS events are not received if a long break sequence is received.
11	—	Reserved, should be cleared. Refer to note 1 below.
12	CCR	Control character received and rejected. Set when a control character is recognized and stored in the receive control character register RCCR.
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. In multidrop mode, the receiver automatically enters hunt mode; otherwise, reception continues when a buffer is available. The latest point that an RxBD can be changed to empty and guarantee avoiding the busy condition is the middle of the stop bit of the first character to be stored in that buffer.
14	TX	Tx event. Set when a buffer is sent. If TxBD[CR] = 1, TX is set no sooner than when the last stop bit of the last character in the buffer begins transmission. If TxBD[CR] = 0, TX is set after the last character is written to the Tx FIFO. TX also represents a $\overline{\text{CTS}}$ lost error; check TxBD[CT].
15	RX	Rx event. Set when a buffer is received, which is no sooner than the middle of the first stop bit of the character that caused the buffer to close. Also represents a general receiver error (overrun, $\overline{\text{CD}}$ lost, parity, idle sequence, and framing errors); the RxBD status and control fields indicate the specific error.

[1] Reserved bits in the SCCE should not be masked in the SCCM register.

20.20 SCC UART Status Register (SCCS)

The SCC UART status register (SCCS), shown in Figure 20-12, monitors the real-time status of RXD.

	0	6	7
Field	—		ID
Reset	0000_0000_0000_0000		
R/W	R		
Addr	0x11A17 (SCCS1); 0x11A57 (SCCS3); 0x11A77 (SCCS4)		

Figure 20-12. SCC Status Register for UART Mode (SCCS)

Table 20-13 describes UART SCCS fields.

Table 20-13. UART SCCS Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	ID	Idle status. Set when RXD has been a logic one for at least a full character time. 0 The line is not idle. 1 The line is idle.

20.21 SCC UART Programming Example

The following initialization sequence is for the 9,600 baud, 8 data bits, no parity, and stop bit of an SCC in UART mode assuming a 66-MHz system frequency. BRG1 and SCC4 are used. The controller is configured with $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$ active; $\overline{\text{CTS4}}$ acts as an automatic flow-control signal.

1. Configure port D pins to enable TXD4 and RXD4. Set PPARD[21,22] and PDIRD[21] and clear PDIRD[22] and PSORD[21,22].
2. Configure ports C and D pins to enable $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$ and $\overline{\text{CD4}}$. Set PPARC[8,9], PPARD[20] and PDIRD[20] and clear PDIRC[8,9], PSORC[8,9] and PSORD[20].
3. Configure BRG1. Write BRGC1 with 0x0001_035A. The DIV16 bit is not used and the divider is 429 (decimal). The resulting BRG1 clock is 16× the preferred bit rate.
4. Connect BRG1 to SCC4 using the CPM mux. Clear CMXSCR[RS4CS,TS4CS].
5. Connect the SCC4 to the NMSI. Clear CMXSCR[SC4].
6. Write RBASE and TBASE in the SCC4 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write 0x0CE1_0000 to CPCR to execute the INIT RX AND TX PARAMS command for SCC4. This command updates RBPTR and TBPTR of the serial channel with the new values of RBASE and TBASE.
8. Write RFCR with 0x10 and TFCR with 0x10 for normal operation.
9. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
10. Write MAX_IDL with 0x0000 in the parameter RAM to disable the maximum idle functionality for this example.
11. Set BRKCR to 0x0001 so STOP TRANSMIT commands send only one break character.
12. Clear PAREC, FRMEC, NOSEC, and BRKEC in parameter RAM.
13. Clear UADDR1 and UADDR2. They are not used.
14. Clear TOSEQ. It is not used.
15. Write CHARACTER1–8 with 0x8000. They are not used.
16. Write RCCM with 0xC0FF. It is not used.
17. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to the RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].

18. Initialize the TxBD. Assume the buffer is at 0x0000_2000 in main memory and contains sixteen 8-bit characters. Write 0xB000 to the TxBD[Status and Control], 0x0010 to TxBD[Data Length], and 0x00002000 to TxBD[Buffer Pointer].
19. Write 0xFFFF to SCCE4 to clear any previous events.
20. Write 0x0003 to SCCM4 to allow the TX and RX interrupts.
21. Write 0x0040_0000 to the SIMR_L so SCC4 can generate a system interrupt. Initialize SIPNR_L by writing 0xFFFF_FFFF to it.
22. Write 0x0000_0020 to GSMR_H4 to configure a small Rx FIFO width.
23. Write 0x0002_8004 to GSMR_L4 to configure 16× sampling for transmit and receive, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits), and the SCC for UART mode. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
24. Set PSMR4 to 0xB000 to configure automatic flow control using $\overline{\text{CTS}}$, 8-bit characters, no parity, 1 stop bit, and asynchronous SCC UART operation.
25. Write 0x0002_8034 to GSMR_L4 to enable the transmitter and receiver. This ensures that ENT and ENR are enabled last.

NOTE

After 16 bytes are sent, the transmit buffer is closed. Additionally, the receive buffer is closed after 16 bytes are received. Data received after 16 bytes causes a busy (out-of-buffers) condition because only one RxBD is prepared.

20.22 S-Records Loader Application

This section describes a downloading application that uses an SCC UART controller. The application performs S-record downloads and uploads between a host computer and an intelligent peripheral through a serial asynchronous line. S-records are strings of ASCII characters that begin with ‘S’ and end in an end-of-line character. This characteristic is used to impose a message structure on the communication between the devices. For flow control, each device can transmit XON and XOFF characters, which are not part of the program being uploaded or downloaded.

For simplicity, assume that the line is not multidrop (no addresses are sent) and that each S-record fits into a single buffer. Follow the basic UART initialization sequence above in Section 20.21, “SCC UART Programming Example,” except allow for more and larger buffers and create the control character table as described in Table 20-14.

Table 20-14. UART Control Characters for S-Records Example

Character	Description
Line feed	Both the E and R bits should be cleared. When an end-of-line character is received, the current buffer is closed and made available to the core for processing. This buffer contains an entire S record that the processor can now check and copy to memory or disk as required.

Table 20-14. UART Control Characters for S-Records Example (continued)

Character	Description
XOFF	E should be cleared; R should be set. Whenever the core receives a control-character-received (CCR) interrupt and the RCCR contains XOFF, the software should immediately stop transmitting by setting PSMR[FRZ]. This keeps the other station from losing data when it runs out of Rx buffers.
XON	XON should be received after XOFF. E should be cleared and R should be set. PSMR[FRZ] on the transmitter should now be cleared. The CPM automatically resumes transmission of the serial line at the point at which it was previously stopped. Like XOFF, the XON character is not stored in the receive buffer.

To receive S-records, the core must wait for an RX interrupt, indicating that a complete S-record buffer was received. Transmission requires assembling S-records into buffers and linking them to the TxBD table; transmission can be paused when an XOFF character is received. This scheme minimizes the number of interrupts the core receives (one per S-record) and relieves it from continually scanning for control characters.

Chapter 21
SCC HDLC Mode

High-level data link control (HDLC) is one of the most common protocols in the data link layer, layer 2 of the OSI model. Many other common layer 2 protocols, such as SDLC, SS#7, AppleTalk, LAPB, and LAPD, are based on HDLC and, in particular, its framing structure. Figure 21-1 shows the HDLC framing structure.

HDLC uses a zero insertion/deletion process (bit-stuffing) to ensure that a data bit pattern matching the delimiter flag does not occur in a field between flags. The HDLC frame is synchronous and relies on the physical layer for clocking and synchronization of the transmitter/receiver.

An address field is needed to carry the frame's destination address because the layer 2 frame can be sent over point-to-point links, broadcast networks, packet-switched or circuit-switched systems. An address field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. SDLC and LAPB use an 8-bit address. SS#7 has no address field because it is always used in point-to-point signaling links. LAPD divides its 16-bit address into different fields to specify various access points within one device. LAPD also defines a broadcast address. Some HDLC-type protocols permit addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends on the protocol using the frame. The length of the data in the data field depends on the frame protocol. Layer 3 frames are carried in this data field. Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16 bits long but can be as long as 32 bits. In HDLC, the lsb of each octet is sent first; the msb of the CRC is sent first.

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. In a non-multiplexed modem interface, SCC outputs connect directly to external pins. Modem signals can be supported through port C. The Rx and Tx clocks can be supplied from either the bank of baud rate generators, by the DPLL, or externally. An SCC can also be connected through the TDM channels of the serial interface (SI). In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.

21.1 SCC HDLC Features

The main features of an SCC in HDLC mode are follows:

- Flexible buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (Rx and Tx)
- Received-frames threshold to reduce interrupt overhead
- Can be used with the SCC DPLL

- Four address comparison registers with mask
- Maintenance of five 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation and checking
- Detection of nonoctet aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- Automatic retransmission in case of collision

21.2 SCC HDLC Channel Frame Transmission

The HDLC transmitter is designed to work with little or no core intervention. Once enabled by the core, a transmitter starts sending flags or idles as programmed in the HDLC mode register (PSMR). The HDLC polls the first BD in the TxBD table. When there is a frame to transmit, the SCC fetches the data (address, control, and information) from the first buffer and starts sending the frame after inserting the minimum number of flags specified between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the SCC appends the CRC and closing flag. In HDLC mode, the lsb of each octet and the msb of the CRC are sent first. Figure 21-1 shows a typical HDLC frame.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 bits	16 bits	8 bits	8*n* bits	16 bits	8 bits

Figure 21-1. HDLC Framing Structure

After a closing flag is sent, the SCC updates the frame status bits of the BD and clears TxBD[R] (buffer ready). At the end of the current buffer, if TxBD[L] is not set (multiple buffers per frame), only TxBD[R] is cleared. Before the SCC proceeds to the next TxBD in the table, an interrupt can be issued if TxBD[I] is set. This interrupt programmability allows the core to intervene after each buffer, after a specific buffer, or after each frame.

The STOP TRANSMIT command can be used to expedite critical data ahead of previously linked buffers or to support efficient error handling. When the SCC receives a STOP TRANSMIT command, it sends idles or flags instead of the current frame until it receives a RESTART TRANSMIT command. The GRACEFUL STOP TRANSMIT command can be used to insert a high-priority frame without aborting the current one—a graceful-stop-complete event is generated in SCCE[GRA] when the current frame is finished. See Section 21.6, “SCC HDLC Commands.”

21.3 SCC HDLC Channel Frame Reception

The HDLC receiver is designed to work with little or no core intervention to perform address recognition, CRC checking, and maximum frame length checking. Received frames can be used to implement any HDLC-based protocol.

Once enabled by the core, the receiver waits for an opening flag character. When it detects the first byte of the frame, the SCC compares the frame address with four user-programmable, 16-bit address registers and an address mask. The SCC compares the received address field with the user-defined values after masking with the address mask. To detect broadcast (all ones) address frames, one address register must be written with all ones.

If an address match is detected, the SCC fetches the next BD and SCC starts transferring the incoming frame to the buffer if it is empty. When the buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. If the incoming frame is larger than the current buffer, the SCC continues receiving using the next BD in the table.

During reception, the SCC checks for frames that are too long (using MFLR). When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. RxBD[Data Length] of the last BD in the HDLC frame contains the entire frame length. This also enables software to identify the frames in which the maximum frame length violations occur. The SCC sets RxBD[L] (last buffer in frame), writes the frame status bits, and clears RxBD[E]. It then generates a maskable event (SCCE[RXF]) to indicate a frame was received. The SCC then waits for a new frame. Back-to-back frames can be received with only one shared flag between frames.

The received frames threshold parameter (RFTHR) can be used to postpone interrupts until a specified number of frames is received. This function can be combined with a timer to implement a timeout if fewer than the specified number of threshold frames is received.

Note that SCCs in HDLC mode, or any other synchronous mode, must receive a minimum of eight clocks after the last bit arrives to account for Rx FIFO delay.

21.4 SCC HDLC Parameter RAM

For HDLC mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 21-1.

Table 21-1. HDLC-Specific SCC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**C_MASK**	Word	CRC mask. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For 32-bit CRC-CCITT, initialize with 0xDEBB_20E3.
0x38	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF.
0x3C	**DISFC**	Hword	Modulo 2^{16} counters maintained by the CP. Initialize them while the channel is disabled. DISFC (Discarded frame counter) Counts error-free frames discarded due to lack of free buffers. CRCEC (CRC error counter) Includes frames not addressed to the user or frames received in the BSY condition, but does not include overrun errors. ABTSC (Abort sequence counter) NMARC (Nonmatching address received counter) Includes error-free frames only. RETRC (Frame retransmission counter) Counts number of frames resent due to collision.
0x3E	**CRCEC**	Hword	
0x40	**ABTSC**	Hword	
0x42	**NMARC**	Hword	
0x44	**RETRC**	Hword	

Table 21-1. HDLC-Specific SCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x46	**MFLR**	Hword	Max frame length register. The HDLC compares the incoming HDLC frame's length with the user-defined limit in MFLR. If the limit is exceeded, the rest of the frame is discarded and RxBD[LG] is set in the last BD of that frame. At the end of the frame the SCC reports frame status and frame length in the last RxBD. The MFLR is defined as all in-frame bytes between the opening and closing flags.
0x48	MAX_CNT	Hword	Maximum length counter. A temporary down-counter used to track frame length.
0x4A	**RFTHR**	Hword	Received frames threshold. Used to reduce potential interrupt overhead when each in a series of short HDLC frames causes an SCCE[RXF] event. Setting RFTHR determines the frequency of RXF interrupts, which occur only when the RFTHR limit is reached. Provide enough empty RxBDs for the number of frames specified in RFTHR.
0x4C	RFCNT	Hword	Received frames count. RFCNT is a down-counter used to implement RFTHR.
0x4E	**HMASK**	Hword	Mask register (HMASK) and four address registers (HADDR*n*) for address recognition. The SCC reads the frame address from the HDLC receiver, compares it with the HADDRs, and masks the result with HMASK. Setting an HMASK bit enables the corresponding comparison bit, clearing a bit masks it. When a match occurs, the frame address and data are written to the buffers. When no match occurs and a frame is error-free, the nonmatching address received counter (NMARC) is incremented. The eight low-order bits of HADDR*n* should contain the first address byte after the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDR*n* should contain 0xAA68 and HMASK should contain 0xFFFF. For 8-bit addresses, clear the eight high-order HMASK bits. See Figure 21-2.
0x50	**HADDR1**	Hword	
0x52	**HADDR2**	Hword	
0x54	**HADDR3**	Hword	
0x56	**HADDR4**	Hword	
0x58	TMP	Hword	Temporary storage.
0x5A	TMP_MB	Hword	Temporary storage.

[1] From SCC base. See Section 19.3.1, "SCC Base Addresses."

Figure 21-2 shows 16- and 8-bit address recognition.

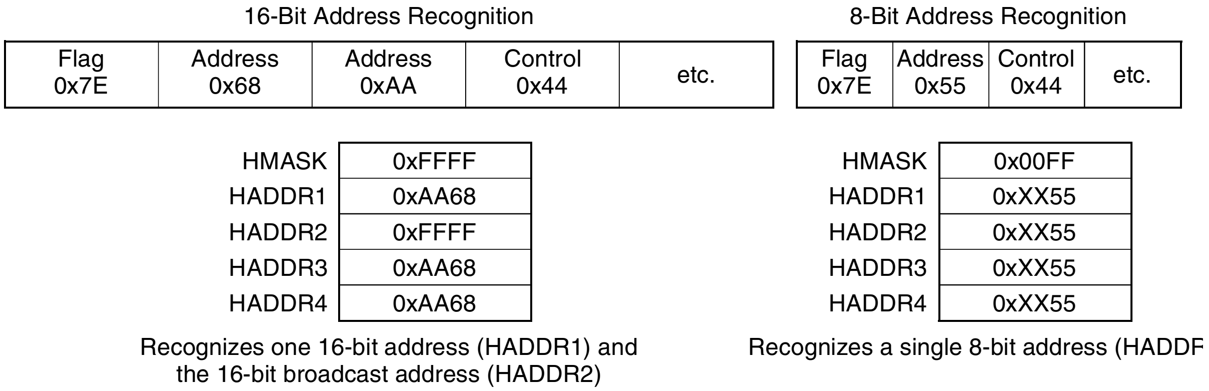


Figure 21-2. HDLC Address Recognition

21.5 Programming the SCC in HDLC Mode

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. The HDLC controller uses the same buffer and BD data structure as other modes and supports multibuffer operation and address comparisons. Receive errors are reported through the RxBD; transmit errors are reported through the TxBD.

21.6 SCC HDLC Commands

The transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 21-2.

Table 21-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 64 Tx clocks, or immediately if TODR[TOD] = 1, and begins sending data if TxBD[R] is set. If the SCC receives the STOP TRANSMIT command while not transmitting, the transmitter stops polling the BDs. If the SCC receives the command during transmission, transmission is aborted after a maximum of 64 additional bits, the Tx FIFO is flushed, and the current BD pointer TBPTR is not advanced (no new BD is accessed). The transmitter then sends an abort sequence (0x7F) and stops polling the BDs. When not transmitting, the channel sends flags or idles as programmed in the GSMR. Note that if PSMR[MFF] = 1, multiple small frames could be flushed from the Tx FIFO; a GRACEFUL STOP TRANSMIT command prevents this.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly. Unlike a STOP TRANSMIT command, it stops transmission after the current frame is finished or immediately if no frame is being sent. SCCE[GRA] is set when transmission stops. HDLC Tx parameters and Tx BDs can then be updated. TBPTR points to the next TxBD. Transmission begins once TxBD[R] of the next BD is set and a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables frames to be sent on the transmit channel. The HDLC controller expects this command after a STOP TRANSMIT is issued and the channel in its GSMR is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. The transmitter resumes from the current BD.
INIT TX PARAMETERS	Resets the Tx parameters in the parameter RAM. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 21-3.

Table 21-3. Receive Commands

Command	Description
ENTER HUNT MODE	After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. While the SCC is looking for the beginning of a frame, that SCC is in hunt mode. The ENTER HUNT MODE command is used to force the HDLC receiver to stop receiving the current frame and enter hunt mode, in which the HDLC continually scans the input data stream for a flag sequence. After receiving the command, the buffer is closed and the CRC is reset. Further frame reception uses the next BD.
CLOSE RXBD	Should not be used in the HDLC protocol
INIT RX PARAMETERS	Resets the Rx parameters in the parameter RAM.; issue only when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

21.7 Handling Errors in the SCC HDLC Controller

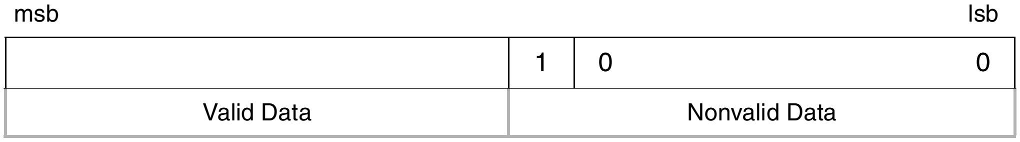
The SCC HDLC controller reports frame reception and transmission errors using BDs, error counters, and the SCCE. Transmission errors are described in Table 21-4.

Table 21-4. Transmit Errors

Error	Description
Transmitter underrun	The channel stops transmitting, closes the buffer, sets TxBD[UN], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is issued. The SCC send and receive FIFOs are 32 bytes each.
$\overline{\text{CTS}}$ lost during frame transmission	The channel stops transmitting, closes the buffer, sets TxBD[CT], and generates the TXE interrupt if not masked. Transmission resumes after a RESTART TRANSMIT command. If this error occurs on the first or second buffer of the frame and PSMR[RTE] = 1, the channel resends the frame when $\overline{\text{CTS}}$ is reasserted and no error is reported. If collisions are possible, to ensure proper retransmission of multi-buffer frames, the first two buffers of each frame should in total contain more than 36 bytes for SCC or 20 bytes for SCC. The channel also increments the retransmission counter RETRC in the parameter RAM.

Reception errors are described in Table 21-5.

Table 21-5. Receive Errors

Error	Description
Overrun	Each SCC maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when a full or partial FIFO's worth of data (according to GSMR_H[RFW]) is received in the Rx FIFO. When an Rx FIFO overrun occurs, the previous byte is overwritten by the next byte. The previous data byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates an RXF interrupt if not masked. The receiver then enters hunt mode. Even if an overrun occurs during a frame whose address is not recognized, an RxBD with data length two is opened to report the overrun and the interrupt is generated.
$\overline{\text{CD}}$ lost during frame reception	Highest priority error. The channel stops frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if not masked. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode.
Abort sequence	Occurs when seven or more consecutive ones are received. When this occurs while receiving a frame, the channel closes the buffer, sets RxBD[AB] and generates a maskable RXF interrupt. The channel also increments the abort sequence counter ABTSC. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode.
Nonoctet aligned frame	The channel writes the received data to the buffer, closes the buffer, sets RxBD[NO], and generates a maskable RXF interrupt. CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. An immediate back-to-back frame is still received. The nonoctet data may be derived from the last word in the buffer as follows:  **Note:** If buffer swapping is used (RFCR[BO] = 0b0x), the figure above refers to the last byte, rather than the last word, of the buffer. The lsb of each octet is sent first while the msb of the CRC is sent first.
CRC	The channel writes the received CRC to the buffer, closes the buffer, sets RxBD[CR], generates a maskable RXF interrupt, and increments the CRC error counter CRCEC. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

21.8 HDLC Mode Register (PSMR)

The protocol-specific mode register (PSMR), shown in Figure 21-3, functions as the HDLC mode register.

	0 3	4 5	6	7	8	9	10	11	12	13 15
Field	NOF	CRC	RTE	—	FSE	DRT	BUS	BRM	MFF	—
Reset	0									
R/W	R/W									
Addr	0x11A08 (PSMR1); 0x11A48 (PSMR3); 0x11A68 (PSMR4)									

Figure 21-3. HDLC Mode Register (PSMR)

Table 21-6 describes PSMR HDLC fields.

Table 21-6. PSMR HDLC Field Descriptions

Bits	Name	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames. If NOF = 0b0000, no flags are inserted between frames and the closing flag of one frame is followed by the opening flag of the next frame in the case of back-to-back frames. NOF can be modified on-the-fly.
4–5	CRC	CRC selection. 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1. x1 Reserved 10 32-bit CCITT-CRC (Ethernet and HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1.
6	RTE	Retransmit enable. 0 No retransmission 1 Automatic frame retransmission is enabled. Particularly useful in the HDLC bus protocol and ISDN applications where multiple HDLC controllers can collide. Note that retransmission occurs only if a lost $\overline{\text{CTS}}$ occurs on the first or second buffer of the frame.
7	—	Reserved, should be cleared
8	FSE	Flag sharing enable. FSE can be set only if GSMR_H[RTSM] is already set. Can be modified on-the-fly. 0 Normal operation 1 If NOF[0–3] = 0b0000, a single shared flag is sent between back-to-back frames. Other values of NOF[0–3] are decremented by 1. Useful in signaling system #7 applications.
9	DRT	Disable receiver while transmitting. 0 Normal operation 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$. This helps if the HDLC channel is on a multidrop line and the SCC does not need to receive its own transmission. **Note:** If DRT = 1, GSMR_H[CDS] should be cleared unless both of the following are true: the same clock is used for TCLK and RCLK, and CTS either has synchronous timing or is always asserted.
10	BUS	HDLC bus mode. 0 Normal HDLC operation 1 HDLC bus operation is selected. See Section 21.15, “HDLC Bus Mode with Collision Detection.”

Table 21-6. PSMR HDLC Field Descriptions (continued)

Bits	Name	Description
11	BRM	HDLC bus $\overline{\text{RTS}}$ mode. Valid only if BUS = 1. Otherwise, it is ignored. 0 Normal $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is asserted on the first bit of the Tx frame and negated after the first collision bit is received. 1 Special $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is delayed by one bit with respect to the normal case, which helps when the HDLC bus protocol is being run locally and sent over a long-distance line at the same time. The one-bit delay allows $\overline{\text{RTS}}$ to be used to enable the transmission line buffers so that the electrical effects of collisions are not sent over the transmission line.
12	MFF	Multiple frames in Tx FIFO. The receiver is not affected. 0 Normal operation. The Tx FIFO must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. 1 The Tx FIFO can hold multiple frames, but lost $\overline{\text{CTS}}$ may not be reported on the buffer/frame it occurred on. This can improve performance of HDLC transmissions of small back-to-back frames or when the number of flags between frames should be limited.
13–15	—	Reserved, should be cleared

21.9 SCC HDLC Receive Buffer Descriptor (RxBD)

The CP uses the RxBD, shown in Figure 21-4, to report on data received for each buffer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	F	**CM**	—	DE	—	LG	NO	AB	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 21-4. SCC HDLC Receive Buffer Descriptor (RxBD)

Table 21-7 describes HDLC RxBD status and control fields.

Table 21-7. SCC HDLC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or reception stopped because of an error. The core can read or write to any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer is not full. The CP controls the BD and buffer. The core should not update the BD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in the RxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE. The number of BDs in this table are programmable and determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 SCCE[RXB] is not set after this buffer is used; SCCE[RXF] is unaffected. 1 SCCE[RXB] or SCCE[RXF] is set when the SCC uses this buffer.

Table 21-7. SCC HDLC RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
4	L	Last buffer in frame. 0 Not the last buffer in frame 1 Last buffer in frame. Indicates reception of a closing flag or an error, in which case one or more of the CD, OV, AB, and LG bits are set. The SCC writes the number of frame octets to the data length field.
5	F	First in frame. 0 Not the first buffer in a frame 1 First buffer in a frame
6	**CM**	Continuous mode. Note that RxBD[E] is cleared if an error occurs during reception, regardless of CM. 0 Normal operation 1 RxBD[E] is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten next time the CP accesses it.
7	—	Reserved, should be cleared
8	DE	DPLL error. Set when a DPLL error occurs while this buffer is being received. DE is also set due to a missing transition when using decoding modes in which a transition is required for every bit. Note that when a DPLL error occurs, the frame closes and error checking halts.
9	—	Reserved, should be cleared
10	LG	Rx frame length violation. Set when a frame larger than the maximum defined for this channel is recognized. Only the maximum-allowed number of bytes (MFLR) is written to the buffer. This event is not reported until the buffer is closed, SCCE[RXF] is set, and the closing flag is received. The total number of bytes received between flags is still written to the data length field.
11	NO	Rx nonoctet aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	AB	Rx abort sequence. Set when at least seven consecutive ones are received during frame reception.
13	CR	Rx CRC error. Set when a frame contains a CRC error. CRC bytes received are always written to the Rx buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost (NMSI mode only). Set when $\overline{\text{CD}}$ is negated during frame reception.

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).” Because HDLC is a frame-based protocol, RxBD[Data Length] of the last buffer of a frame contains the total number of frame bytes, including the 2 or 4 bytes for CRC. Figure 21-5 shows an example of how RxBDs are used in receiving.

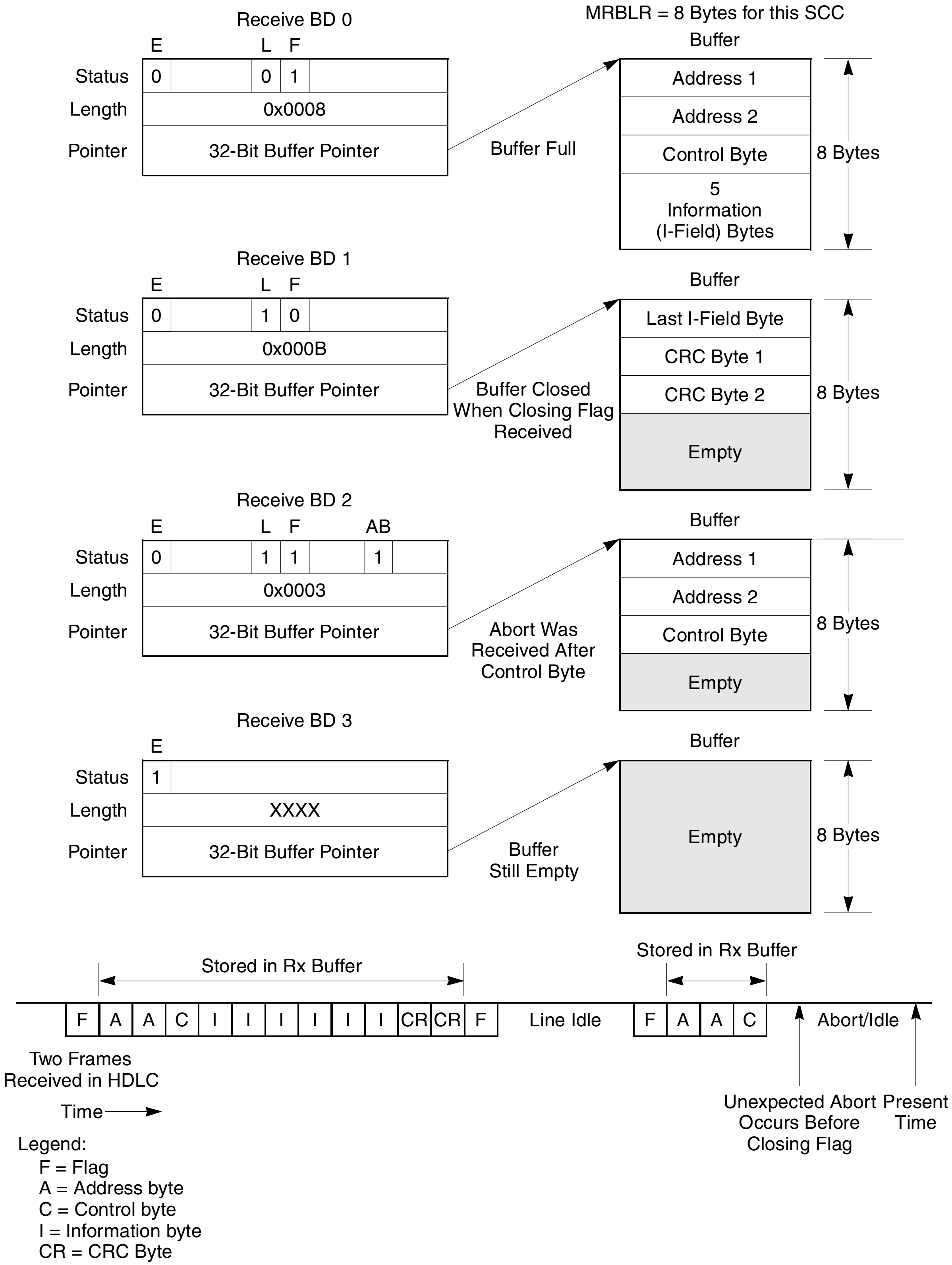


Figure 21-5. SCC HDLC Receiving Using RxBDs

21.10 SCC HDLC Transmit Buffer Descriptor (TxBD)

The CP uses the TxBD, shown in Figure 21-6, to confirm transmissions and indicate error conditions.

	0	1	2	3	4	5	6	7 – 13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	—	UN	CT
Offset + 2	**Data Length**									
Offset + 4	**Tx Buffer Pointer**									
Offset + 6										

Figure 21-6. SCC HDLC Transmit Buffer Descriptor (TxBD)

Table 21-8 describes HDLC TxBD status and control fields.

Table 21-8. SCC HDLC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. Both the buffer and the BD can be updated. The CP clears R after the buffer is sent or an error is encountered. 1 The buffer has not been sent or is being sent and the BD cannot be updated.
1	—	Reserved, should be cleared
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table 1 Last BD in the BD table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined by TxBD[W] and the space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 SCCE[TXB] is not set after this buffer is used; SCCE[TXE] is unaffected. 1 SCCE[TXB] or SCCE[TXE] is set when this buffer is processed, causing interrupts if not masked.
4	**L**	Last. 0 Not the last buffer in the frame 1 Last buffer in the frame
5	**TC**	Tx CRC. Valid only when TxBD[L] = 1. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear TxBD[R] after this BD is closed allowing the buffer to be resent the next time the CP accesses this BD. However, TxBD[R] is cleared if an error occurs during transmission, regardless of CM.
7–13	—	Reserved, should be cleared
14	UN	Underrun. Set after the SCC sends a buffer and a transmitter underrun occurred.
15	CT	$\overline{\text{CTS}}$ lost. Indicates when $\overline{\text{CTS}}$ in NMSI mode or layer 1 grant is lost in GCI or IDL mode during frame transmission. If data from more than one buffer is currently in the FIFO when this error occurs, the HDLC writes CT in the current BD after sending the buffer.

The data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).”

21.11 HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

The SCC event register (SCCE) is used as the HDLC event register to report events recognized by the HDLC channel and to generate interrupts. When an event is recognized, the SCC sets the corresponding SCCE bit. Interrupts generated through SCCE can be masked in the SCC mask register (SCCM), which has the same bit format as the SCCE. Setting an SCCM bit enables the corresponding interrupt; clearing a bit masks it. SCCE bits are cleared by writing ones; writing zeros has no effect. All unmasked bits must be cleared before the CP clears the internal interrupt request. Figure 21-7 shows SCCE/SCCM for HDLC operation.

	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—				DCC	FLG	IDL	GRA	—		TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x11A10 (SCCE1); 0x11A50 (SCCE3); 0x11A70 (SCCE4) 0x11A14 (SCCM1); 0x11A54 (SCCM3); 0x11A74 (SCCM4)														

Figure 21-7. HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

Table 21-9 describes SCCE/SCCM fields.

Table 21-9. SCCE/SCCM Field Descriptions [1]

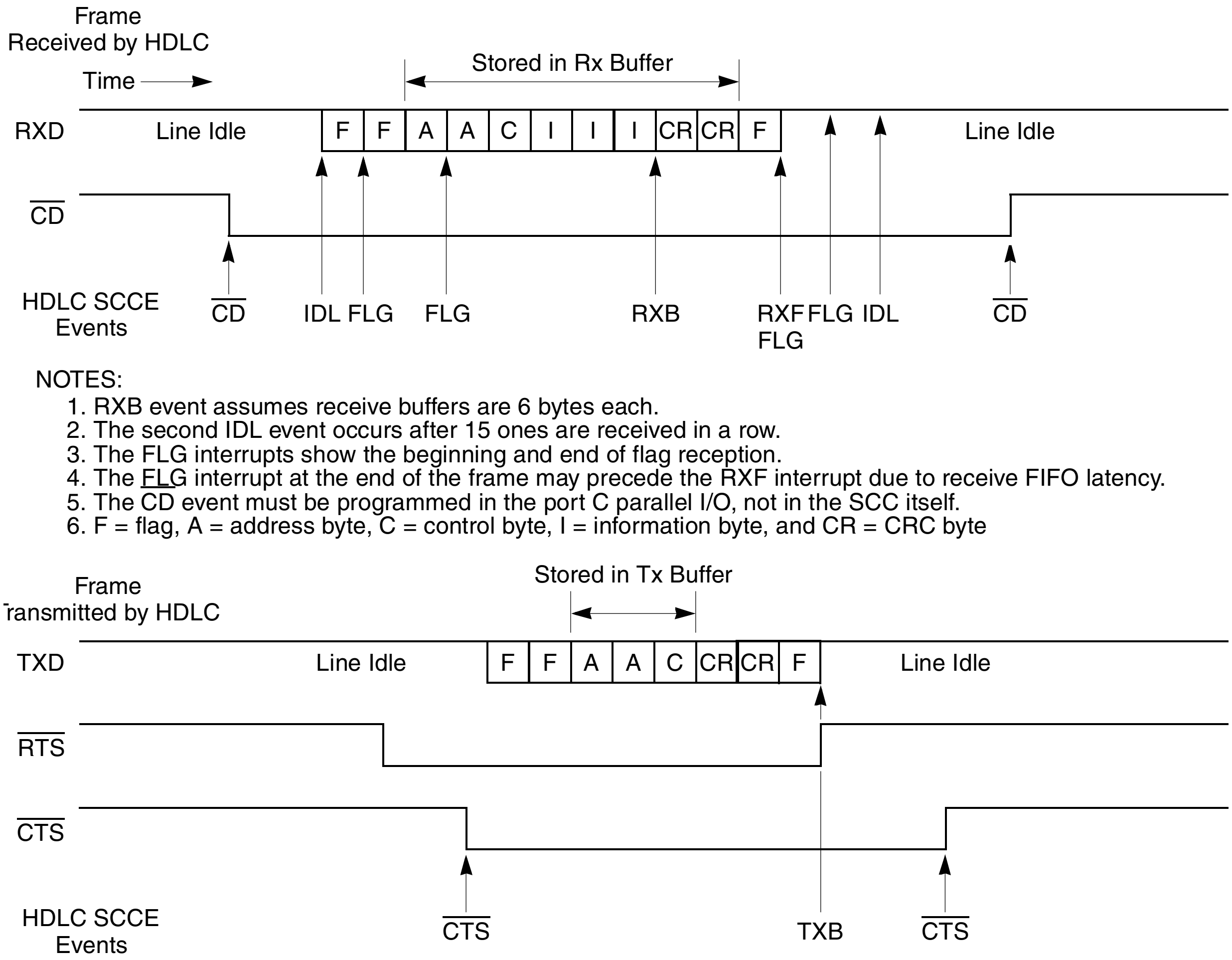
Bits	Nam e	Description
0–4	—	Reserved, should be cleared. Refer to note 1 below.
5	DCC	DPLL carrier sense changed. Set when the carrier sense status generated by the DPLL changes. Real-time status can be read in SCCS[CS]. This is not the $\overline{CD}$ status reported in port C. Valid only when the DPLL is used.
6	FLG	Flag status. Set when the SCC stops or starts receiving HDLC flags. Real-time status can be read in SCCS[FG].
7	IDL	Idle sequence status changed. Set when HDLC line status changes. Real-time status of the line can be read in SCCS[ID].
8	GRA	Graceful stop complete. A GRACEFUL STOP TRANSMIT command completed execution. Set as soon as the transmitter has sent a frame in progress when the command was issued. Set immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared. Refer to note 1 below.
11	TXE	Tx error. Indicates an error ($\overline{CTS}$ lost or underrun) has occurred on the transmitter channel. This event is not maskable via the TxBD[I] bit.
12	RXF	Rx frame. Set when the number of receive frames specified in RFTHR are received on the HDLC channel. It is set no sooner than two clocks after the last bit of the closing flag is received. This event is not maskable through the RxBD[I] bit.
13	BSY	Busy condition. Indicates a frame arrived but was discarded due to a lack of buffers.

Table 21-9. SCCE/SCCM Field Descriptions (continued)[1]

Bits	Name	Description
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. TXB is set when a buffer is sent on the HDLC channel. For the last buffer in the frame, TXB is not set before the last bit of the closing flag begins its transmission; otherwise, it is set after the last byte of the buffer is written to the Tx FIFO.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when the HDLC channel receives a buffer that is not the last in a frame.

[1] Reserved bits in the SCCE should not be masked in the SCCM register.

Figure 21-8 shows interrupts that can be generated using the HDLC protocol.



NOTES:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.
3. The FLG interrupts show the beginning and end of flag reception.
4. The FLG interrupt at the end of the frame may precede the RXF interrupt due to receive FIFO latency.
5. The CD event must be programmed in the port C parallel I/O, not in the SCC itself.
6. F = flag, A = address byte, C = control byte, I = information byte, and CR = CRC byte

NOTES:
1. TXB event shown assumes all three bytes were put into a single buffer.
2. Example shows one additional opening flag. This is programmable.
3. The CTS event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 21-8. SCC HDLC Interrupt Event Example

21.12 SCC HDLC Status Register (SCCS)

The SCC status register (SCCS), shown in Figure 21-9, permits monitoring of real-time status conditions on RXD. The real-time status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are part of the port C parallel I/O.

	0 — 4	5	6	7
Field	—	FG	CS	ID
Reset	0000_0000			
R/W	R			
Addr	0x11A17 (SCCS1); 0x11A57 (SCCS3); 0x11A77 (SCCS4)			

Figure 21-9. CC HDLC Status Register (SCCS)

Table 21-10 describes HDLC SCCS fields.

Table 21-10. HDLC SCCS Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. The line is checked after the data has been decoded by the DPLL. 0 HDLC flags are not being received. The most recently received 8 bits are examined every bit time to see if a flag is present. 1 HDLC flags are being received. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once it is set, it remains set at least 8 bit times and the next eight received bits are examined. If another flag occurs, FG stays set for at least another eight bits. If not, it is cleared and the search begins again.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	ID	Idle status. 0 The line is busy. 1 Set when RXD is a logic 1 (idle) for 15 or more consecutive bit times. It is cleared after a single logic 0 is received.

21.13 SCC HDLC Programming Examples

The following sections show examples for programming SCCs in HDLC mode. The first example uses an external clock. The second example implements Manchester encoding.

21.14 SCC HDLC Programming Example #1

The following initialization sequence is for an SCC HDLC channel with an external clock. SCC4 is used with $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$ active; CLK5 is used for both the HDLC receiver and transmitter.

1. Configure port D pins to enable TXD4 and RXD4. Set PPARD[21,22] and PDIRD[21] and clear PDIRD[22] and PSORD[21,22].
2. Configure ports C and D pins to enable RTS4, CTS4 and CD4. Set PPARC[8,9], PPARD[20] and PDIRD[20] and clear PDIRC[8,9], PSORC[8,9] and PSORD[20].

3. Configure port C pin 27 to enable the CLK5 pin. Set PPARC[27] and clear PDIRC[27] and PSORC[27].
4. Connect CLK5 to SCC4 using the CPM mux. Write 0b100 to CMXSCR[R4CS] and CMXSCR[T4CS].
5. Connect the SCC4 to the NMSI (its own set of pins). clear CMXSCR[SC4].
6. Write RBASE and TBASE in the SCC4 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM and one TxBD following it, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write RBASE and TBASE in the SCC4 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM and one TxBD following it, write RBASE with 0x0000 and TBASE with 0x0008.
8. Write 0x0CE1_0000 to CPCR to execute the INIT RX AND TX PARAMS command for SCC4. This command updates RBPTR and TBPTR of the serial channel with the new values of RBASE and TBASE.
9. Write RFCR with 0x10 and TFCR with 0x10 for normal operation.
10. Write MRBLR with the maximum number of bytes per Rx buffer. Choose 256 bytes (MRBLR = 0x0100) so an entire Rx frame can fit in one buffer.
11. Write C_MASK with 0x0000F0B8 to comply with 16-bit CCITT-CRC.
12. Write C_PRES with 0x0000FFFF to comply with 16-bit CCITT-CRC.
13. Clear DISFC, CRCEC, ABTSC, NMARC, and RETRC for clarity.
14. Write MFLR with 0x0100 so the maximum frame size is 256 bytes.
15. Write RFTHR with 0x0001 to allow interrupts after each frame.
16. Write HMASK with 0x0000 to allow all addresses to be recognized.
17. Clear HADDR1–HADDR4 for clarity.
18. Initialize the RxBD. Assume the buffer is at 0x0000_1000 in main memory. RxBD[Status and Control]= 0xB000, RxBD[Data Length] = 0x0000 (not required), and RxBD[Buffer Pointer] = 0x0000_1000.
19. Initialize the TxBD. Assume the Tx data frame is at 0x0000_2000 in main memory and contains five 8-bit characters. TxBD[Status and Control] = 0xBC00, TxBD[Data Length] = 0x0005, and TxBD[Buffer Pointer] = 0x0000_2000.
20. Write 0xFFFF to SCCE to clear any previous events.
21. Write 0x001A to SCCM to enable TXE, RXF, and TXB interrupts.
22. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
23. Write 0x0000_0000 to GSMR_H4 to enable normal $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ behavior with idles (not flags) between frames.
24. Write 0x0000_0000 to GSMR_L4 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to control transmission and reception in HDLC mode. Normal Tx clock operation is used. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled. If inverted HDLC operation is preferred, set RINV and TINV.

25. Write 0x0000 to PSMR4 to configure one opening and one closing flag, 16-bit CCITT-CRC, and prevent multiple frames in the FIFO.
26. Write 0x00000030 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last.

NOTE

After 5 bytes and CRC have been sent, the Tx buffer is closed; the Rx buffer is closed after a frame is received. Frames larger than 256 bytes cause a busy (out-of-buffers) condition because only one RxBD is prepared.

21.14.1 SCC HDLC Programming Example #2

The following sequence initializes an HDLC channel that uses the DPLL in a Manchester encoding. Provide a clock that is 16× the chosen bit rate of CLK5. Then connect CLK5 to the HDLC transmitter and receiver. (A baud rate generator could be used instead.) Configure SCC4 to use $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$.

1. Follow steps 1–22 in example #1 above.
2. Write 0x004A_A400 to GSMR_L4 to make carrier sense always active, a 16-bit preamble of ‘01’ patterns, 16× operation of the DPLL and Manchester encoding for the receiver and transmitter, and HDLC mode. $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ should be configured to control transmission and reception. Do not set GSMR[ENT, ENR].
3. Write 0x0000 to PSMR4 to use one opening and one closing flag and 16-bit CCITT-CRC and to reject multiple frames in the FIFO.
4. Write 0x004A_A430 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write to GSMR_L4 ensures that ENT and ENR are enabled last.

21.15 HDLC Bus Mode with Collision Detection

The HDLC controller includes an option for hardware collision detection and retransmission on an open-drain connected HDLC bus, referred to as HDLC bus mode. Most HDLC-based controllers provide only point-to-point communications; however, the HDLC bus enhancement allows implementation of an HDLC-based LAN and other point-to-multipoint configurations. The HDLC bus is based on techniques used in the CCITT ISDN I.430 and ANSI T1.605 standards for D-channel point-to-multipoint operation over the S/T interface. However, the HDLC bus does not fully comply with I.430 or T1.605 and cannot replace devices that implement these protocols. Instead, it is more suited to non-ISDN LAN and point-to-multipoint configurations.

Review the basic features of the I.430 and T1.605 before learning about the HDLC bus. The I.430 and T1.605 define a way to connect eight terminals over the D-channel of the S/T ISDN bus. The layer 2 protocol is a variant of HDLC, called LAPD. However, at layer 1, a method is provided to allow the eight terminals to send frames to the switch through the physical S/T bus.

To determine whether a channel is clear, the S/T interface device looks at an echo bit on the line designed to echo the last bit sent on the D channel. Depending on the class of terminal and the context, an S/T interface device waits for 7–10 ones on the echo bit before letting the LAPD frame begin transmission, after which the S/T interface monitors transmitted data. As long as the echo bit matches the sent data, transmission continues. If the echo bit is ever 0 when the transmit bit is 1, a collision occurs between terminals; the station(s) that sent a zero stops transmitting. The station that sent a 1 continues as normal.

The I.430 and T1.605 standards provide a physical layer protocol that allows multiple terminals to share one physical connection. These protocols handle collisions efficiently because one station can always complete its transmission, at which point, it lowers its own priority to give other devices fair access to the physical connection.

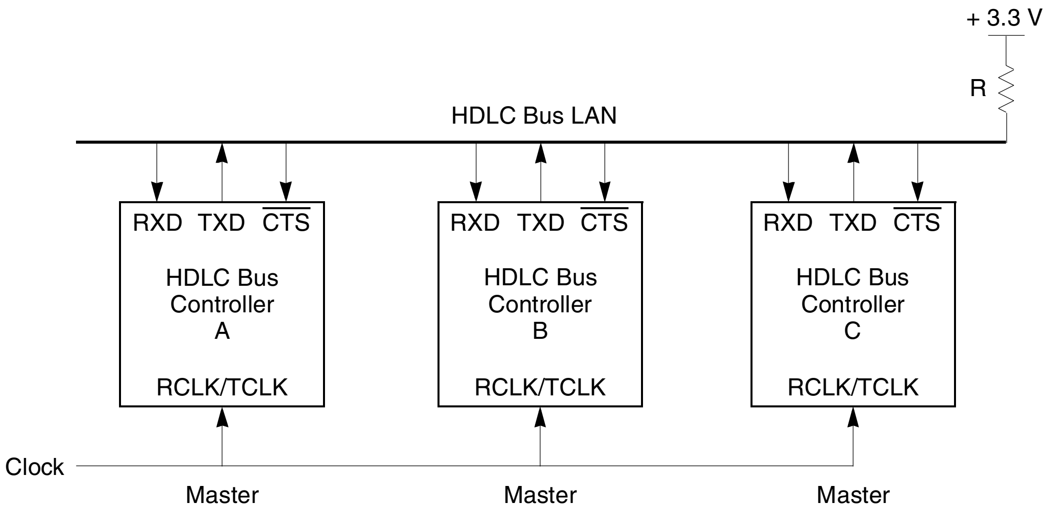
The HDLC bus differs from the I.430 and T1.605 standards as follows:

- The HDLC bus uses a separate input signal rather than the echo bit to monitor data; the transmitted data is simply connected to the $\overline{\text{CTS}}$ input.
- The HDLC bus is a synchronous, digital open-drain connection for short-distance configurations, rather than the more complex S/T interface.
- Any HDLC-based frame protocol can be used at layer 2, not just LAPD.
- HDLC bus devices wait 8–10 rather than 7–10 bit times before transmitting. (HDLC bus has only one class.)

The collision-detection mechanism supports only the following:

- NRZ-encoded data
- A common synchronous clock for all receivers and transmitters
- Non-inverted data (GSMR[RINV, TINV] = 0)
- Open-drain connection with no external transceivers

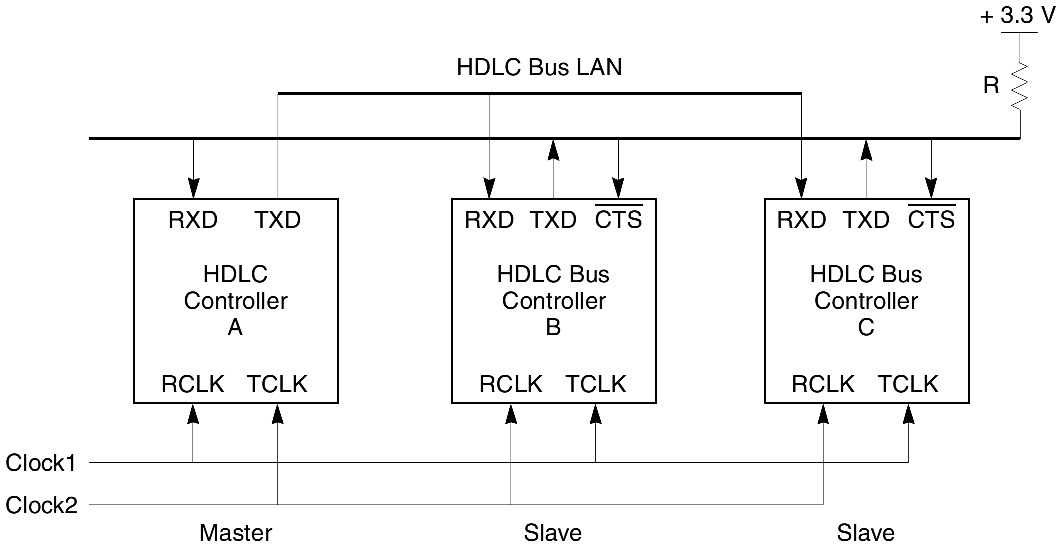
Figure 21-10 shows the most common HDLC bus LAN configuration, a multiple master configuration. A station can transfer data to or from any other LAN station. Transmissions are half duplex, which is typical in LANs.



NOTES:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock is a common RCLK/TCLK for all stations.

Figure 21-10. Typical HDLC Bus Multiple Master Configuration

In single-master configuration, a master station transmits to any slave station without collisions. Slaves communicate only with the master, but can experience collisions in their access over the bus. In this configuration, a slave that communicates with another slave must first transmit its data to the master, where the data is buffered in RAM and then resent to the other slave. The benefit of this configuration, however, is that full-duplex operation can be obtained. In a point-to-multipoint environment, this is the preferred configuration. Figure 21-11 shows the single-master configuration.



NOTES:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock1 is the master RCLK and the slave TCLK.
4. Clock2 is the master TCLK and the slave RCLK.

Figure 21-11. Typical HDLC Bus Single-Master Configuration

21.15.1 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

21.15.2 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 21-12 shows how $\overline{\text{CTS}}$ is used to detect collisions.

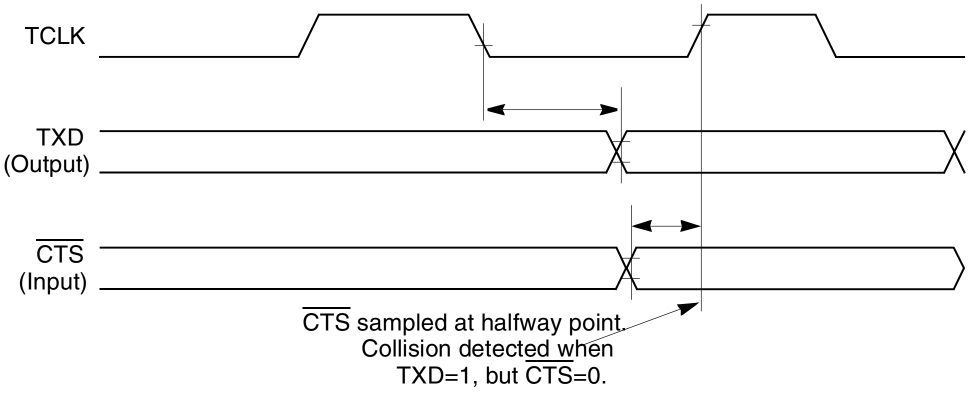

TCLK
TXD
(Output)
$\overline{\text{CTS}}$
(Input)
$\overline{\text{CTS}}$ sampled at halfway point.
Collision detected when
TXD=1, but $\overline{\text{CTS}}$=0.

Figure 21-12. Detecting an HDLC Bus Collision

If both the destination address and source address are included in the HDLC frame, then a predefined priority of stations results; if two stations begin to transmit simultaneously, they necessarily detect a collision no later than the end of the source address.

The HDLC bus priority mechanism ensures that stations share the bus equally. To minimize idle time between messages, a station normally waits for eight one bits on the line before attempting transmission. After successfully sending a frame, a station waits for 10 rather than eight consecutive one bits before attempting another transmission. This mechanism ensures that another station waiting to transmit acquires the bus before a station can transmit twice. When a low priority station detects 10 consecutive ones, it tries to transmit; if it fails, it reinstates the high priority of waiting for only eight ones.

21.15.3 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 21-13.

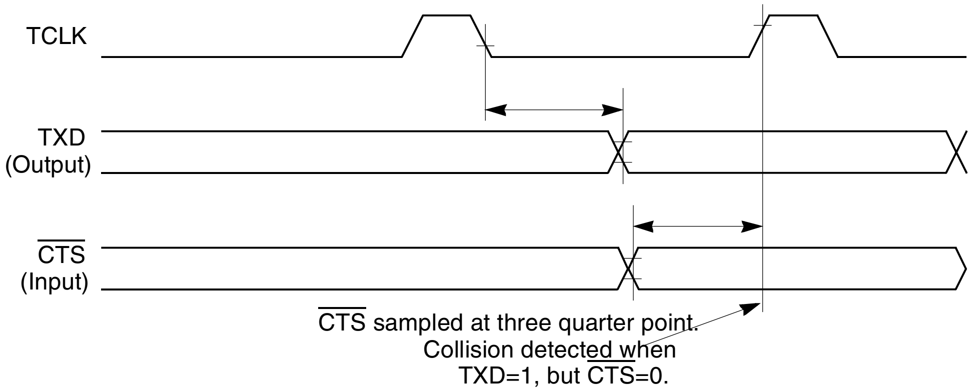
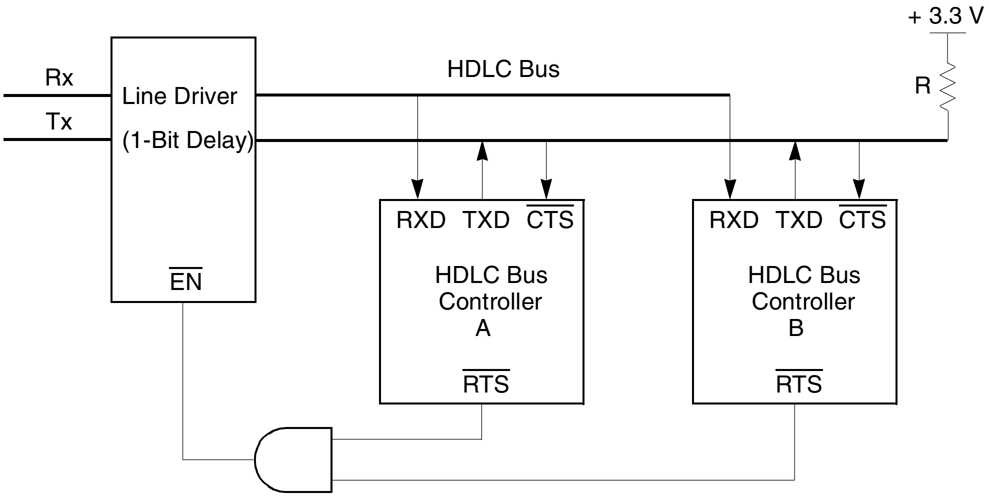


Figure 21-13. Nonsymmetrical Tx Clock Duty Cycle for Increased Performance

21.15.4 Delayed $\overline{\text{RTS}}$ Mode

Figure 21-14 shows local HDLC bus controllers using a standard transmission line. 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 21-14. HDLC Bus Transmission Line Configuration

Normally, $\overline{\text{RTS}}$ goes active at the beginning of the opening flag's first bit. Setting PSMR[BRM] delays $\overline{\text{RTS}}$ by one bit, which is useful when the HDLC bus connects multiple local stations to a transmission line. If the transmission line driver has a one-bit delay, the delayed $\overline{\text{RTS}}$ can be used to enable the output of the line driver. As a result, the electrical effects of collisions are isolated locally. Figure 21-15 shows $\overline{\text{RTS}}$ timing.

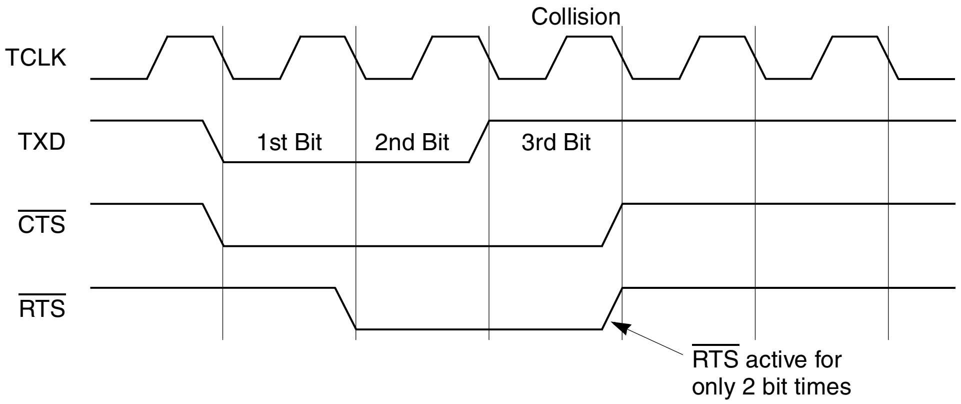
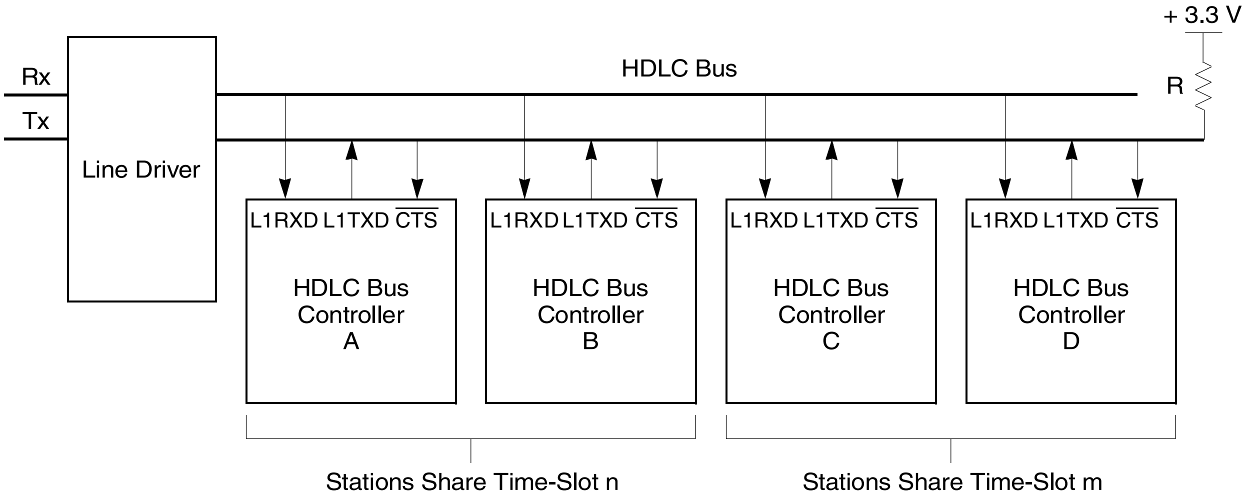


Figure 21-15. Delayed $\overline{\text{RTS}}$ Mode

21.15.5 Using the Time-Slot Assigner (TSA)

HDLC bus controllers can be used with a time-division multiplexed transmission line, as shown in Figure 21-16. Local stations use time slots to communicate over the TDM transmission line; stations that share a time slot use the HDLC bus protocol to control access to the bus.


+ 3.3 V
HDLC Bus
R
Rx
Tx
Line Driver
L1RXD L1TXD $\overline{\text{CTS}}$
HDLC Bus Controller A
L1RXD L1TXD $\overline{\text{CTS}}$
HDLC Bus Controller B
L1RXD L1TXD $\overline{\text{CTS}}$
HDLC Bus Controller C
L1RXD L1TXD $\overline{\text{CTS}}$
HDLC Bus Controller D
Stations Share Time-Slot n
Stations Share Time-Slot m

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 21-16. HDLC Bus TDM Transmission Line Configuration

The local SCCs in HDLC bus mode communicate only with the transmission line and not with each other. The SCCs use the TSA of the serial interface, receiving and sending data over L1TXD*x* and L1RXD*x*. Because collisions are still detected from the individual SCC $\overline{\text{CTS}}$ pin, it must be configured to connect to the chosen SCC. Because the SCC only receives clocks during its time slot, $\overline{\text{CTS}}$ is sampled only during the Tx clock edges of the particular SCC time slot.

21.15.6 HDLC Bus Protocol Programming

The HDLC bus on the MPC8272 is implemented using the SCC in HDLC mode with bus-specific options selected in the PSMR and GSMR, as outlined below. See also Section 21.5, “Programming the SCC in HDLC Mode.”

21.15.6.1 Programming GSMR and PSMR for the HDLC Bus Protocol

To program the protocol-specific mode register (PSMR), set the bits as described below:

- Configure NOF as preferred
- Set RTE and BUS to 1
- Set BRM to 1 if delayed $\overline{\text{RTS}}$ is desired
- Configure CRC to 16-bit CRC CCITT (0b00).
- Configure other bits to zero or default.

To program the general SCC mode register (GSMR), set the bits as described below:

- Set MODE to HDLC mode (0b0000).
- Configure CTSS to 1 and all other bits to zero or default.
- Configure the DIAG bits for normal operation (0b00).
- Configure RDCR and TDCR for 1× clock (0b00).
- Configure TENC and RENC for NRZ (0b000).
- Clear RTSM to send idles between frames.
- Set GSMR_L[ENT, ENR] as the last step to begin operation.

21.15.6.2 HDLC Bus Controller Programming Example

Except for the above discussion in Section 21.15.6.1, “Programming GSMR and PSMR for the HDLC Bus Protocol,” use the example in Section 21.14, “SCC HDLC Programming Example #1.”

Chapter 22
SCC BISYNC Mode

The byte-oriented BISYNC protocol was developed by IBM for use in networking products. There are three classes of BISYNC frames—transparent, nontransparent with header, and nontransparent without header, shown in Figure 22-1. The transparent frame type in BISYNC is not related to transparent mode, discussed in Chapter 23, “SCC Transparent Mode.” Transparent BISYNC mode allows full binary data to be sent with any possible character pattern. Each class of frame starts with a standard two-octet synchronization pattern and ends with a block check code (BCC). The end-of-text character (ETX) is used to separate the text and BCC fields.

Nontransparent with Header

SYN1	SYN2	SOH	Header	STX	Text	ETX	BCC

Nontransparent without Header

SYN1	SYN2	STX	Text	ETX	BCC

Transparent

SYN1	SYN2	DLE	STX	Transparent Text	DLE	ETX	BCC

Figure 22-1. Classes of BISYNC Frames

The bulk of a frame is divided into fields whose meaning depends on the frame type. The BCC is a 16-bit CRC format if 8-bit characters are used; it is a combination longitudinal (sum check) and vertical (parity) redundancy check if 7-bit characters are used. In transparent operation, a special character (DLE) is defined that tells the receiver that the next character is text, allowing BISYNC control characters to be valid text data in a frame. A DLE sent as data must be preceded by a DLE character. This is sometimes called byte-stuffing. The physical layer of the BISYNC communications link must synchronize the receiver and transmitter, usually by sending at least one pair of synchronization characters before each frame.

BISYNC protocol is unusual in that a transmit underrun need not be an error. If an underrun occurs, a synchronization pattern is sent until data is again ready. In nontransparent operation, the receiver discards additional synchronization characters (SYNCs) as they are received. In transparent mode, DLE-SYNC pairs are discarded. Normally, for proper transmission, an underrun must not occur between the DLE and its following character. This failure mode cannot occur with the MPC8272.

An SCC can be configured as a BISYNC controller to handle basic BISYNC protocol in normal and transparent modes. The controller can work with the time-slot assigner (TSA) or non-multiplexed serial interface (NMSI). The controller has separate transmit and receive sections whose operations are asynchronous with the core and either synchronous or asynchronous with other SCCs.

22.1 Features

The following list summarizes features of the SCC in BISYNC mode:

- Flexible data buffers
- Eight control character recognition registers
- Automatic SYNC1–SYNC2 detection
- 16-bit pattern (bisync)
- 8-bit pattern (monosync)
- 4-bit pattern (nibblesync)
- External SYNC pin support
- SYNC/DLE stripping and insertion
- CRC16 and LRC (sum check) generation/checking
- VRC (parity) generation/checking
- Supports BISYNC transparent operation
- Maintains parity error counter
- Reverse data mode capability

22.2 SCC BISYNC Channel Frame Transmission

The BISYNC transmitter is designed to work with almost no core intervention. When the transmitter is enabled, it starts sending SYN1–SYN2 pairs in the data synchronization register (DSR) or idles as programmed in the PSMR. The BISYNC controller polls the first BD in the channel’s TxBD table. If there is a message to send, the controller fetches the message from memory and starts sending it after the SYN1–SYN2 pair. The entire pair is always sent, regardless of GSMR[SYNL].

After a buffer is sent, if the last (TxBD[L]) and the Tx block check sequence (TxBD[TB]) bits are set, the BISYNC controller appends the CRC16/LRC and then writes the message status bits in TxBD status and control fields and clears the ready bit, TxBD[R]. It then starts sending the SYN1–SYN2 pairs or idles, according to GSMR[RTSM]. If the end of the current BD is reached and TxBD[L] is not set, only TxBD[R] is cleared. In both cases, an interrupt is issued according to TxBD[I]. TxBD[I] controls whether interrupts are generated after transmission of each buffer, a specific buffer, or each block. The controller then proceeds to the next BD.

If no additional buffers have been sent to the controller for transmission, an in-frame underrun is detected and the controller starts sending syncs or idles. If the controller is in transparent mode, it sends DLE-sync pairs. Characters are included in the block check sequence (BCS) calculation on a per-buffer basis. Each buffer can be programmed independently to be included or excluded from the BCS calculation; thus, excluded characters must reside in a separate buffer. The controller can reset the BCS generator before sending a specific buffer. In transparent mode, the controller inserts a DLE before sending a DLE character, so that only one DLE is used in the calculation.

22.3 SCC BISYNC Channel Frame Reception

Although the receiver is designed to work with almost no core intervention, the user can intervene on a per-byte basis if necessary. The receiver performs CRC16, longitudinal (LRC) or vertical redundancy (VRC) checking, sync stripping in normal mode, DLE-sync stripping, stripping of the first DLE in DLE-DLE pairs in transparent mode, and control character recognition. Control characters are discussed in Section 22.6, “SCC BISYNC Control Character Recognition.”

When enabled, the receiver enters hunt mode where the data is shifted into the receiver shift register one bit at a time and the contents of the shift register are compared to the contents of DSR[SYN1, SYN2]. If the two are unequal, the next bit is shifted in and the comparison is repeated. When registers match, hunt mode is terminated and character assembly begins. The controller is character-synchronized and performs SYNC stripping and message reception. It reverts to hunt mode when it receives an ENTER HUNT MODE command, an error condition, or an appropriate control character.

When receiving data, the controller updates the CR bit in the BD for each byte transferred. When the buffer is full, the controller clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the buffer length, the controller fetches the next BD; if E is zero, reception continues to its buffer.

When a BCS is received, it is checked and written to the buffer. The BISYNC controller sets the last bit, writes the message status bits into the BD, clears the E bit, and then generates a maskable interrupt, indicating that a block of data was received and is in memory. The BCS calculations do not include SYNCs (in nontransparent mode) or DLE-SYNC pairs (in transparent mode).

Note that GSMR_H[RFW] should be set for an 8-bit-wide receive FIFO for the BISYNC receiver. See Section 19.1.1, “The General SCC Mode Registers (GSMR1–GSMR4).”

22.4 SCC BISYNC Parameter RAM

For BISYNC mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 22-1.

Table 22-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**CRCC**	Word	CRC constant temp value.
0x38	**PRCRC**	Hword	Preset receiver/transmitter CRC16/LRC. These values should be preset to all ones or zeros, depending on the BCS used.
0x3A	**PTCRC**	Hword	
0x3C	**PAREC**	Hword	Receive parity error counter. This 16-bit (modulo 2^{16}) counter maintained by the CP counts parity errors on receive if the parity feature of BISYNC is enabled. Initialize PAREC while the channel is disabled.
0x3E	**BSYNC**	Hword	BISYNC SYNC register. Contains the value of the SYNC to be sent as the second byte of a DLE–SYNC pair in an underrun condition and stripped from incoming data on receive once the receiver synchronizes to the data using the DSR and SYN1–SYN2 pair. See Section 22.7, “BISYNC SYNC Register (BSYNC).”

Table 22-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x40	**BDLE**	Hword	BISYNC DLE register. Contains the value to be sent as the first byte of a DLE–SYNC pair and stripped on receive. See Section 22.8, "SCC BISYNC DLE Register (BDLE)."
0x42	**CHARACTER1**	Hword	Control character 1–8. These values represent control characters that the BISYNC controller recognizes. See Section 22.6, "SCC BISYNC Control Character Recognition."
0x44	**CHARACTER2**	Hword	
0x46	**CHARACTER3**	Hword	
0x48	**CHARACTER4**	Hword	
0x4A	**CHARACTER5**	Hword	
0x4C	**CHARACTER6**	Hword	
0x4E	**CHARACTER7**	Hword	
0x50	**CHARACTER8**	Hword	
0x52	**RCCM**	Hword	Receive control character mask. Masks CHARACTER*n* comparison so control character classes can be defined. Setting a bit enables and clearing a bit masks comparison. See Section 22.6, "SCC BISYNC Control Character Recognition."

[1] From SCC*x* base address. See Section 19.3.1, "SCC Base Addresses."

GSMR[MODE] determines the protocol for each SCC. The SYN1–SYN2 synchronization characters are programmed in the DSR (see Section 19.1.3, "Data Synchronization Register (DSR).") The BISYNC controller uses the same basic data structure as other modes; receive and transmit errors are reported through their respective BDs. There are two basic ways to handle BISYNC channels:

- The controller inspects the data on a per-byte basis and interrupts the core each time a byte is received.
- The controller can be programmed so software handles the first two or three bytes. The controller directly handles subsequent data without interrupting the core.

22.5 SCC BISYNC Commands

Transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 22-2.

Table 22-2. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 transmit clocks. This command stops transmission after a maximum of 64 additional bits without waiting for the end of the buffer and the transmit FIFO to be flushed. TBPTR is not advanced, no new BD is accessed, and no new buffers are sent for this channel. SYNC–SYNC or DLE–SYNC pairs are sent continually until a RESTART TRANSMIT is issued. A STOP TRANSMIT can be used when an EOT sequence should be sent and transmission should stop. After transmission resumes, the EOT sequence should be the first buffer sent to the controller. Note that the controller remains in transparent or normal mode after it receives a STOP TRANSMIT or RESTART TRANSMIT command.
GRACEFUL STOP TRANSMIT	Stops transmission after the current frame finishes sending or immediately if there is no frame being sent. SCCE[GRA] is set once transmission stops. Then BISYNC transmit parameters and TxBDs can be modified. The TBPTR points to the next TxBD. Transmission resumes when the R bit of the next BD is set and a RESTART TRANSMIT is issued.
RESTART TRANSMIT	Lets characters be sent on the transmit channel. The BISYNC controller expects it after a STOP TRANSMIT or a GRACEFUL STOP TRANSMIT command is issued, after a transmitter error occurs, or after a STOP TRANSMIT is issued and the channel is disabled in its SCCM. The controller resumes transmission from the current TBPTR in the channel's TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel's parameter RAM to their reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets transmit and receive parameters.

Receive commands are described in Table 22-3.

Table 22-3. Receive Commands

Command	Description
RESET BCS CALCULATION	Immediately resets the receive BCS accumulator. It can be used to reset the BCS after recognizing a control character, thus signifying that a new block is beginning.
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled in SCCM, the channel is in receive enable mode and uses the first BD. This command forces the controller to stop receiving and enter hunt mode, during which the controller continually scans the data stream for an SYN1–SYN2 sequence as programmed in the DSR. After receiving the command, the current receive buffer is closed and the BCS is reset. Message reception continues using the next BD.
CLOSE RXBD	Used to force the SCC to close the current RxBD if it is in use and to use the next BD for subsequent data. If data is not being received, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel's parameter RAM to reset state. Issue only when the receiver is disabled. An INIT TX AND RX PARAMETERS resets transmit and receive parameters.

22.6 SCC BISYNC Control Character Recognition

The BISYNC controller recognizes special control characters that customize the protocol implemented by the BISYNC controller and aid its operation in a DMA-oriented environment. They are used for receive buffers longer than one byte. In single-byte buffers, each byte can be easily inspected so control character recognition should be disabled.

The control character table 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 22-2.

Offset from SCC*x* Base	0	1	2	3 7	8 15
0x42	E	B	H	—	CHARACTER1
0x44	E	B	H	—	CHARACTER2
0x46	E	B	H	—	CHARACTER3
0x48	E	B	H	—	CHARACTER4
0x4A	E	B	H	—	CHARACTER5
0x4D	E	B	H	—	CHARACTER6
0x4E	E	B	H	—	CHARACTER7
0x50	E	B	H	—	CHARACTER8
0x52	1	1	1	—	MASK VALUE(RCCM)

Figure 22-2. Control Character Table and RCCM

Table 22-4 describes control character table and RCCM fields.

Table 22-4. Control Character Table and RCCM Field Descriptions

Offset	Bit	Name	Description
0x42– 0x50	0	E	End of table. 0 This entry is valid. The lower eight bits are checked against the incoming character. In tables with eight control characters, E should be zero in all eight positions. 1 The entry is not valid. No other valid entries exist beyond this entry.
	1	B	BCS expected. A maskable interrupt is generated after the buffer is closed. 0 The character is written into the receive buffer and the buffer is closed. 1 The character is written into the receive buffer. The receiver waits for one LRC or two CRC bytes of BCS and then closes the buffer. This should be used for ETB, ETX, and ITB.
	2	H	Hunt mode. Enables hunt mode when the current buffer is closed. 0 The BISYNC controller maintains character synchronization after closing this buffer. 1 The BISYNC controller enters hunt mode after closing the buffer. When the B bit is set, the controller enters hunt mode after receiving the BCS.
	3–7	—	Reserved
	8–15	CHARACTER*n*	Control character 1–8. When using 7-bit characters with parity, include the parity bit in the character value.
0x52	0–2	—	All ones
	3–7	—	Reserved
	8–15	RCCM	Received control character mask. Masks comparison of CHARACTER*n*. Each bit of RCCM masks the corresponding bit of CHARACTER*n*. 0 Mask this bit in the comparison of the incoming character and CHARACTER*n*. 1 The address comparison on this bit proceeds normally and no masking occurs. If RCCM is not set, erratic operation can occur during control character recognition.

22.7 BISYNC SYNC Register (BSYNC)

The BSYNC register, shown in Figure 22-3, defines BISYNC stripping and SYNC character insertion. When an underrun occurs, the BISYNC controller inserts SYNC characters until the next buffer is available for transmission. If the receiver is not in hunt mode when a SYNC character is received, it discards this character if the valid bit (BSYNC[V]) is set. When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

	0	1	2	3	4	5	6	7	8	15
Field	V	DIS	0	0	0	0	0	0	SYNC	
Reset	Undefined									
R/W	R/W									
Addr	SCC Base + 0x3E									

Figure 22-3. BISYNC SYNC (BSYNC)

Table 22-5 describes BSYNC fields.

Table 22-5. BSYNC Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable BSYNC stripping 0 Normal mode 1 BSYNC stripping disabled (BISYNC transparent mode only)
2–7	—	All zeros
8–15	SYNC	SYNC character

22.8 SCC BISYNC DLE Register (BDLE)

Seen in Figure 22-4, the BDLE register is used to define the BISYNC stripping and insertion of DLE characters. When an underrun occurs while a message is being sent in transparent mode, the BISYNC controller inserts DLE-SYNC pairs until the next buffer is available for transmission.

In transparent mode, the receiver discards any DLE character received and excludes it from the BCS if the valid bit (BDLE[V]) is set. If the second character is SYNC, the controller discards it and excludes it from the BCS. If it is a DLE, the controller writes it to the buffer and includes it in the BCS. If it is not a DLE or SYNC, the controller examines the control character table and acts accordingly. If the character is not in the table, the buffer is closed with the DLE follow character error bit set. If the valid bit is not set, the receiver treats the character as a normal character. When using 7-bit characters with parity, the parity bit should be included in the DLE register value.

	0	1	2	3	4	5	6	7	8	15
Field	V	DIS	0	0	0	0	0	0	DLE	
Reset	Undefined									
R/W	R/W									
Addr	SCC Base + 0x40									

Figure 22-4. BISYNC DLE (BDLE)

Table 22-6 describes BDLE fields.

Table 22-6. BDLE Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable DLE stripping 0 Normal mode 1 DLE stripping disabled. When DIS is enabled in BDLE and on BSYNC the following cases occur: DLE-DLE sequence. Both characters are written to the memory. The BCS is calculated only on the second DLE. DLE-SYNC sequence. Both characters are written to the memory, but neither are included in the BCS calculation. DLE-ETX, DLE-ITB, DLE-ETB sequence, both characters are written to memory. The BCS is calculated only on the second character.
2–7	—	All zeros
8–15	DLE	DLE character

22.9 Sending and Receiving the Synchronization Sequence

The BISYNC channel can be programmed to send and receive a synchronization pattern defined in the DSR. GSMR_H[SYNL] defines pattern length, as shown in Table 22-7. The receiver synchronizes on this pattern. Unless SYNL is zero (external sync), the transmitter always sends the entire DSR contents, lsb first, before each frame—the chosen 4- or 8-bit pattern can be repeated in the lower-order bits.

Table 22-7. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR.															
01	4-Bit															
10	8-Bit															
11	16-Bit															

22.10 Handling Errors in the SCC BISYNC

The controller reports message transmit and receive errors using the channel BDs, error counters, and the SCCE. Modem lines can be directly monitored through the parallel port pins. Table 22-8 describes transmit errors.

Table 22-8. **Transmit Errors**

Error	Description
Transmitter underrun	The channel stops sending the buffer, closes it, sets TxBD[UN], and generates aTXE interrupt if it is enabled. The channel resumes transmission after a RESTART TRANSMIT command is received. Underrun cannot occur between frames or during a DLE–*XXX* pair in transparent mode.
$\overline{\text{CTS}}$ lost during message transmission	The channel stops sending the buffer, closes it, sets TxBD[CT], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is received.

Table 22-9 describes receive errors.

Table 22-9. **Receive Errors**

Error	Description
Overrun	The controller maintains a receiver FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when the first byte is received in the Rx FIFO. If an Rx FIFO overrun occurs, the controller writes the received byte over the previously received byte. The previous character and its status bits are lost. The channel then closes the buffer, sets RxBD[OV], and generates the RXB interrupt if it is enabled. Finally, the receiver enters hunt mode.
$\overline{\text{CD}}$ lost during message reception	The channel stops receiving, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if not masked. This error has the highest priority. If the rest of the message is lost, no other errors are checked in the message. The receiver immediately enters hunt mode.
Parity	The channel writes the received character to the buffer and sets RxBD[PR]. The channel stops receiving, closes the buffer, sets RxBD[PR], and generates the RXB interrupt if it is enabled. The channel also increments PAREC and the receiver immediately enters hunt mode.
CRC	The channel updates the CR bit in the BD every time a character is received with a byte delay of 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.

22.11 BISYNC Mode Register (PSMR)

The PSMR is used as the BISYNC mode register, shown in Figure 22-5. PSMR[RBCS, RTR, RPM, TPM] can be modified on-the-fly.

<table>
<tr><th></th><th>0</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Field</td><td colspan="2">NOS</td><td colspan="2">CRC</td><td>RBCS</td><td>RTR</td><td>RVD</td><td>DRT</td><td colspan="2">—</td><td colspan="2">RPM</td><td colspan="2">TPM</td></tr>
<tr><td>Reset</td><td colspan="14">0</td></tr>
<tr><td>R/W</td><td colspan="14">R/W</td></tr>
<tr><td>Addr</td><td colspan="14">0x (PSMR1); 0x11A48 (PSMR3); 0x11A68 (PSMR4)</td></tr>
</table>

Figure 22-5. Protocol-Specific Mode Register for BISYNC (PSMR)

Table 22-10 describes PSMR fields.

Table 22-10. PSMR Field Descriptions

Bits	Name	Description
0–3	NOS	Minimum number of SYN1–SYN2 pairs (defined in DSR) sent between or before messages. If NOS = 0000, one pair is sent. If NOS = 1111, 16 pairs are sent. The entire pair is always sent, regardless of how GSMR[SYNL] is set. NOS can be modified on-the-fly.
4–5	CRC	CRC selection. x0 Reserved 01 CRC16 (BISYNC). X16 + X15 + X2 + 1. PRCRC and PTCRC should be initialized to all zeros or all ones before the channel is enabled. In either case, the transmitter sends the calculated CRC noninverted and the receiver checks the CRC against zero. Eight-bit data characters (without parity) are configured when CRC16 is chosen. 11 LRC (sum check). (BISYNC). For even LRC, initialize PRCRC and PTCRC to zeros before the channel is enabled; for odd LRC, they should be initialized to ones. Note 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 eight serial clock delay to allow examination of a received byte to determine whether it should used in BCS calculation. 0 Disable receive BCS 1 Enable receive BCS. Should be set (or reset) within the time taken to receive the following data byte. When RBCS is reset, BCS calculations exclude the latest fully received data byte. When RBCS is set, BCS calculations continue as normal.
7	RTR	Receiver transparent mode. 0 Normal receiver mode with SYNC stripping and control character recognition 1 Transparent receiver mode. SYNCs, DLEs, and control characters are recognized only after a leading DLE character. The receiver calculates the CRC16 sequence even if it is programmed to LRC while in transparent mode. Initialize PRCRC to the CRC16 preset value before setting RTR.
8	RVD	Reverse data. 0 Normal operation 1 Any portion of this SCC defined to operate in BISYNC mode operates by reversing the character bit order and sending the msb first.
9	DRT	Disable receiver while sending. DRT should not be set for typical BISYNC operation. 0 Normal operation 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$ signal. This helps if the BISYNC channel is being configured onto a multidrop line and the user does not want to receive its own transmission. Although BISYNC usually uses a half-duplex protocol, the receiver is not actually disabled during transmission. **Note:** If DRT = 1, GSMR_H[CDS] should be cleared unless both of the following are true: the same clock is used for TCLK and RCLK, and CTS either has synchronous timing or is always asserted.
10–11	—	Reserved, should be cleared

Table 22-10. PSMR Field Descriptions (continued)

Bits	Name	Description
12–13	RPM	Receiver parity mode. Selects the type of parity check that the receiver performs. RPM can be modified on-the-fly and is ignored unless CRC = 11 (LRC). Receive parity errors cannot be disabled but can be ignored. 00 Odd parity. The transmitter counts ones in the data word. If the sum is not odd, the parity bit is set to ensure an odd number. An even sum indicates a transmission error. 01 Low parity. If the parity bit is not low, a parity error is reported. 10 Even parity. An even number must result from the calculation performed at both ends of the line. 11 High parity. If the parity bit is not high, a parity error is reported.
14–15	TPM	Transmitter parity mode. Selects the type of parity the transmitter performs and can be modified on-the-fly. TPM is ignored unless CRC = 11 (LRC). 00 Odd parity 01 Force low parity (always send a zero in the parity bit position). 10 Even parity 11 Force high parity (always send a one in the parity bit position).

22.12 SCC BISYNC Receive BD (RxBD)

The CP uses BDs to report on each buffer received. It closes the buffer, generates a maskable interrupt, and starts receiving data into the next buffer after any of the following:

- A user-defined control character is received.
- An error is detected.
- A full receive buffer is detected.
- The ENTER HUNT MODE command is issued.
- The CLOSE RX BD command is issued.

Figure 22-6 shows the SCC BISYNC RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	C	B	**CM**	—	DE	—		DL	PR	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 22-6. SCC BISYNC RxBD

Table 22-11 describes SCC BISYNC RxBD status and control fields.

Table 22-11. SCC BISYNC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty. 0 The buffer is full or stopped receiving because of an error. The core can read or write any fields of this RxBD. The CP does not use this BD as long as the E bit is zero. 1 The buffer is not full. The CP controls this BD and buffer. The core should not update this BD.
1	—	Reserved, should be cleared

Table 22-11. SCC BISYNC RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
2	**W**	Wrap (last BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to. The number of BDs in this table is determined by the W bit and by overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is used. 1 SCCE[RXB] is set when the controller closes this buffer, which can cause an interrupt if it is enabled.
4	C	Control character. The last byte in the buffer is a user-defined control character. 0 The last byte of this buffer does not contain a control character. 1 The last byte of this buffer contains a control character.
5	B	BCS received. The last bytes in the buffer contain the received BCS. 0 This buffer does not contain the BCS. 1 This buffer contains the BCS. A control character may also reside one byte prior to this BCS.
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear E after this BD is closed; the buffer is overwritten when the CP accesses this BD next. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared
8	DE	DPLL error. Set when a DPLL error occurs during reception. In decoding modes where a transition is should occur every bit, the DPLL error is set when a transition is missing.
9–10	—	Reserved, should be cleared
11	DL	DLE follow character error. While in transparent mode, a DLE character was received, and the next character was not DLE, SYNC, or a valid entry in the control characters table.
12	PR	Parity error. Set when a character with parity error is received. Upon a parity error, the buffer is closed; thus, the corrupted character is the last byte of the buffer. A new Rx buffer receives subsequent data.
13	CR	BCS error. Updated every time a byte is written to the buffer. The CR bit includes the calculation for the current byte. By clearing PSMR[RCBS] within eight serial clocks, the user can exclude the current character from the message BCS calculation. The data length field may be read to determine the current character's position.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. Indicates when the carrier detect signal, $\overline{\text{CD}}$, is negated during frame reception.

Data length and buffer pointer fields are described in Section 19.2, "SCC Buffer Descriptors (BDs)." Data length represents the number of octets the CP writes into this buffer, including the BCS. For BISYNC mode, clear these bits. It is incremented each time a received character is written to the buffer.

22.13 SCC BISYNC Transmit BD (TxBD)

The CP arranges data to be sent on an SCC channel in buffers referenced by the channel TxBD table. The CP uses BDs to confirm transmission or indicate errors so the core knows buffers have been serviced. The user configures status and control bits before transmission, but the CP sets them after the buffer is sent.

	0	1	2	3	4	5	6	7	8	9	10	11	13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	**TB**	**CM**	**BR**	**TD**	**TR**	**B**	—		UN	CT
Offset + 2	**Data Length**														
Offset + 4	**Tx Data Buffer Pointer**														
Offset + 6															

Figure 22-7. SCC BISYNC Transmit BD (TxBD)

Table 22-12 describes SCC BISYNC TxBD status and control fields.

Table 22-12. SCC BISYNC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. The current BD and buffer can be updated. The CP clears R after the buffer is sent or after an error condition. 1 The user-prepared buffer has not been sent or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-ported RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is serviced; SCCE[TXE] is unaffected. 1 SCCE[TXB] or SCCE[TXE] is set after the CP services this buffer, which can cause an interrupt.
4	**L**	Last in message. 0 The last character in the buffer is not the last character in the current block. 1 The last character in the buffer is the last character in the current block. The transmitter enters and stays in normal mode after sending the last character in the buffer and the BCS, if enabled.
5	**TB**	Transmit BCS. Valid only when the L bit is set. 0 Send an SYN1–SYN2 or idle sequence (specified in GSMR[RTSM]) after the last character in the buffer. 1 Send the BCS sequence after the last character. The controller also resets the BCS generator after sending the BCS.
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear R after this BD is closed, so the buffer is resent when the CP next accesses this BD. However, R is cleared if an error occurs during transmission, regardless of how CM is set.

Table 22-12. SCC 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 an underrun condition occurs. 1 The transmitter enters or stays in transparent mode after sending the buffer. It automatically inserts DLE–SYNC pairs if an underrun occurs (the controller finishes a buffer with L = 0 and the next BD is not available). It also checks all characters before sending them. If a DLE is detected, another DLE is sent automatically. Insert a DLE or program the controller to insert one before each control character. The transmitter calculates the CRC16 BCS even if PSMR[BCS] is programmed to LRC. Initialize PTCRC to CRC16 before setting TR.
10	**B**	BCS enable. 0 The buffer consists of characters that are excluded from BCS accumulation. 1 The buffer consists of characters that are included in BCS accumulation.
11–13	—	Reserved, should be cleared
14	UN	Underrun. Set when the BISYNC controller encounters a transmitter underrun error while sending the associated data buffer. The CPM writes UN after it sends the associated buffer.
15	CT	$\overline{\text{CTS}}$ lost. The CP sets CT when $\overline{\text{CTS}}$ is lost during message transmission after it sends the data buffer.

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).” Although it is never modified by the CP, data length should be greater than zero. The CPM writes these fields after it finishes sending the buffer.

22.14 BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

The BISYNC controller uses the SCC event register (SCCE) to report events recognized by the BISYNC channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the BISYNC mask register (SCCM). SCCE bits are reset by writing ones; writing zeros has no effect. Unmasked bits must be reset before the CP negates the internal interrupt request signal.

	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—				DCC	—		GRA	—		TXE	RCH	BSY	TXB	RXB
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x11A10 (SCCE1); 0x11A50 (SCCE3); 0x11A70 (SCCE4) 0x11A14 (SCCM1); 0x11A54 (SCCM3); 0x11A74 (SCCM4)														

Figure 22-8. BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

Table 22-13 describes SCCE and SCCM fields.

Table 22-13. SCCE/SCCM Field Descriptions[1]

Bits	Name	Description
0–4	—	Reserved, should be cleared. Refer to note 1 below.
5	DCC	DPLL CS changed. Set when carrier sense status generated by the DPLL changes. Real-time status can be found in SCCS. This is not the $\overline{\text{CD}}$ status discussed elsewhere. Valid only when DPLL is used.
6–7	—	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. This event is not maskable via the TxBD[I] bit.
12	RCH	Receive character. Set when a character is received and written to the buffer.
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. The receiver resumes reception after an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set when a buffer is sent. TXB is set as the last bit of data or the BCS begins transmission.
15	RXB	Rx buffer. Set when the CPM closes the receive buffer on the BISYNC channel.

[1] Reserved bits in the SCCE should not be masked in the SCCM register.

22.15 SCC Status Registers (SCCS)

The SCC status (SCCS) register, seen in Figure 22-9, allows real-time monitoring of RXD. The real-time status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are part of the parallel I/O.

	0	5	6	7
Field	—		CS	—
Reset	0000_0000			
R/W	R			
Addr	0x11A17 (SCCS1); 0x11A57 (SCCS3); 0x11A77 (SCCS4)			

Figure 22-9. SCC Status Registers (SCCS)

Table 22-14 describes SCCS fields.

Table 22-14. SCCS Field Descriptions

Bit	Name	Description
0–5	—	Reserved, should be cleared
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared

22.16 Programming the SCC BISYNC Controller

Software has two ways to handle data received by the BISYNC controller. The simplest is to allocate single-byte receive buffers, request an interrupt on reception of each buffer, and implement BISYNC protocol entirely in software on a byte-by-byte basis. This flexible approach can be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is to prepare and link multi-byte buffers in the RxBD table and use software to analyze the first two to three bytes of the buffer to determine the type of block received. When this is determined, reception continues without further software intervention until it encounters a control character, which signifies the end of the block and causes software to revert to byte-by-byte reception.

To accomplish this, set SCCM[RCH] to enable an interrupt on every received byte so software can analyze each byte. After analyzing the initial characters of a block, either set PSMR[RTR] or issue a RESET BCS CALCULATION command. For example, if a DLE-STX is received, enter transparent mode. By setting the appropriate PSMR bit, the controller strips the leading DLE from DLE-character sequences. Thus, control characters are recognized only when they follow a DLE character. PSMR[RTR] should be cleared after a DLE-ETX is received.

Alternatively, after an SOH is received, a RESET BCS CALCULATION should be issued to exclude SOH from BCS accumulation and reset the BCS. Notice that PSMR[RBCS] is not needed because the controller automatically excludes SYNCs and leading DLEs.

After the type of block is recognized, SCCE[RCH] should be masked. The core does not interrupt data reception until the end of the current block, which is indicated by the reception of a control character matching the one in the receive control character table. Using Table 22-15, the control character table should be set to recognize the end of the block.

Table 22-15. Control Characters

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1

Table 22-15. Control Characters (continued)

ENQ	0	0	0
Next entry	0	X	X

After ETX, a BCS is expected; then the buffer should be closed. Hunt mode should be entered when a line turnaround occurs. ENQ characters are used to stop sending a block and to designate the end of the block for a receiver, but no CRC is expected. After control character reception, set SCCM[RCH] to reenable interrupts for each byte of data received.

22.17 SCC BISYNC Programming Example

This BISYNC controller initialization example for SCC4 uses an external clock. The controller is configured with RTS4, CTS4, and CD4 active. Both the receiver and transmitter use CLK5.

1. Configure port D pins to enable TXD4 and RXD4. Set PPARD[21,22] and PDIRD[21] and clear PDIRD[22] and PSORD[21,22].
2. Configure ports C and D pins to enable RTS4, CTS4 and CD4. Set PPARC[8,9], PPARD[20] and PDIRD[20] and clear PDIRC[8,9], PSORC[8,9], and PSORD[20].
3. Configure port C pin 27 to enable the CLK5 pin. Set PPARC[27] and clear PDIRC[27] and PSORC[27].
4. Connect CLK5 to SCC4 using the CPM mux. Write 0b100 to CMXSCR[R4CS] and CMXSCR[T4CS].
5. Connect the SCC4 to the NMSI (its own set of pins). Clear CMXSCR[SC4].
6. Assuming one RxBD at the beginning of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write 0x0CE1_0000 to CPCR to execute INIT RX AND TX PARAMETERS. This updates RBPTR and TBPTR to the new values of RBASE and TBASE.
8. Write RFCR and TFCR with 0x10 for normal operation.
9. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
10. Write PRCRC with 0x0000 to comply with CRC16.
11. Write PTCRC with 0x0000 to comply with CRC16.
12. Clear PAREC for clarity.
13. Write BSYNC with 0x8033, assuming a SYNC value of 0x33.
14. Write DSR with 0x3333.
15. Write BDLE with 0x8055, assuming a DLE value of 0x55.
16. Write CHARACTER1 with 0x6077, assuming ETX = 0x77.
17. Write CHARACTER2–8 with 0x8000. They are not used.
18. Write RCCM with 0xE0FF. It is not used.

19. Initialize the RxBD and assume the data buffer is at 0x00001000 in main memory. Then write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x00001000 to RxBD[Buffer Pointer].
20. Initialize the TxBD and assume the Tx data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Then write 0xBD20 to TxBD[Status and Control] 0x0005 to TxBD[Data Length], and 0x00002000 to TxBD[Buffer Pointer]. Note that ETX character should be written at the end of user's data.
21. Write 0xFFFF to SCCE to clear any previous events.
22. Write 0x0013 to SCCM to enable the TXE, TXB, and RXB interrupts.
23. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
24. Write 0x0000002C to GSMR_H4 to configure a small receive FIFO width.
25. Write 0x00000008 to GSMR_L4 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits) and the BISYNC mode. Notice that the transmitter (ENT) and receiver (ENR) are not yet enabled.
26. Set PSMR to 0x0600 to configure CRC16, CRC checking on receive, and normal operation (not transparent).
27. Write 0x00000038 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last.

Write 0x00000038 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last. After 5 bytes are sent, the TxBD is closed. The buffer is closed after 16 bytes are received. Any received data beyond 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

Chapter 23
SCC Transparent Mode

Transparent mode (also called totally transparent or promiscuous mode) provides a clear channel on which the SCC can send or receive serial data without bit-level manipulation. Software implements protocols run over transparent mode. An SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.

Transparent mode can be used for serially moving data that requires no superimposed protocol and for applications that require serial-to-parallel and parallel-to-serial conversion for communication among chips on the same board or applications that require data to be switched without interfering with the protocol encoding itself, such as when data from a high-speed time-multiplexed serial stream is multiplexed into low-speed data streams. The concept is to switch the data path without altering the protocol encoded on that data path.

Transparent mode is configured in the GSMR; see Section 19.1.1, "The General SCC Mode Registers (GSMR1–GSMR4)." Transparent mode is selected in GSMR_H[TTX, TRX] for the transmitter and receiver, respectively. Setting both bits enables full-duplex transparent operation. If only one is set, the other half of the SCC uses the protocol specified in GSMR_L[MODE]. This allows loop-back modes to DMA data from one memory location to another while data is converted to a specific serial format.

The SCC operations are asynchronous with the core and can be synchronous or asynchronous with other SCCs. Each clock can be supplied from the internal baud rate generator bank, DPLL output, or external pins.

The SCC can work with the time-slot assigner (TSA) or non-multiplexed serial interface (NMSI) and supports modem lines with the general-purpose I/O pins. Data can be transferred either the msb or lsb first in each octet.

23.1 Features

The following list summarizes the main features of the SCC in transparent mode:

- Flexible buffers
- Automatic SYNC detection on receive
- CRCs can be sent and received
- Reverse data mode
- Another protocol can be performed on the other half of the SCC.
- MC68360-compatible SYNC options

23.2 SCC Transparent Channel Frame Transmission Process

The transparent transmitter is designed to work with almost no intervention from the core. When the core enables the SCC transmitter in transparent mode, the transmitter starts sending idles, which are logic high or encoded ones, as programmed in GSMR_L[TEND]. The SCC polls the first BD in the TxBD table. When there is a message to send, the SCC fetches data from memory, loads the transmit FIFO, and waits for transmitter synchronization, which is achieved with $\overline{\text{CTS}}$ or by waiting for the receiver to achieve synchronization, depending on GSMR_H[TXSY]. Transmission begins when transmitter synchronization is achieved.

When all BD data has been sent, if TxBD[L] is set, the SCC writes the message status bits into the BD, clears TxBD[R], and sends idles until the next BD is ready. If it is ready, some idles are still sent. The transmitter resumes sending only after it achieves synchronization.

If TxBD[L] is cleared when the end of the BD is reached, only TxBD[R] is cleared and the transmitter moves immediately to the next buffer to begin transmission with no gap on the serial line between buffers. Failure to provide the next buffer in time causes a transmit underrun that sets SCCE[TXE].

In both cases, an interrupt is issued according to TxBD[I]. By appropriately setting TxBD[I] in each BD, interrupts are generated after each buffer or group of buffers is sent. The SCC then proceeds to the next BD in the table and any whole number of bytes can be sent. If GSMR_H[REVD] is set, the bit order of each byte is reversed before being sent; the msb of each octet is sent first.

Setting GSMR_H[TFL] makes the transmit FIFO smaller and reduces transmitter latency, but it can cause transmitter underruns at higher transmission speeds. An optional CRC, selected in GSMR_H[TCRC], can be appended to each transparent frame if it is enabled in the TxBD.

When the time-slot assigner (TSA) is used with a transparent-mode channel, synchronization is provided by the TSA. There is a start-up delay for the transmitter, but delays will always be some whole number of complete TSA frames. This means that *n*-byte transmit buffers can be mapped directly into *n*-byte time slots in the TSA frames.

23.3 SCC Transparent Channel Frame Reception Process

When the core enables the SCC receiver in transparent mode, it waits to achieve synchronization before data is received. The receiver can be synchronized to the data by a synchronization pulse or SYNC pattern.

After a buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. It moves to the next RxBD in the table and begins moving data to its buffer. If the next buffer is not available, SCCE[BSY] signifies a busy signal that can generate a maskable interrupt. The receiver reverts to hunt mode when an ENTER HUNT MODE command or an error is received. If GSMR_H[REVD] is set, the bit order of each byte is reversed before it is written to memory.

Setting GSMR_H[RFW] reduces receiver latency by making the receive FIFO smaller, which may cause receiver overruns at higher transmission speeds. The receiver always checks the CRC of the received frame, according to GSMR_H[TCRC]. If a CRC is not required, resulting errors can be ignored.

23.4 Achieving Synchronization in Transparent Mode

Once the SCC transmitter is enabled for transparent operation, the TxBD is prepared and the transmit FIFO is preloaded by the SDMA channel, another process must occur before data can be sent. It is called transmit synchronization. Similarly, once the SCC receiver is enabled for transparent operation in the GSMR and the RxBD is made empty for the SCC, receive synchronization must occur before data can be received. An in-line synchronization pattern or an external synchronization signal can provide bit-level control of the synchronization process when sending or receiving.

23.4.1 Synchronization in NMSI Mode

This section describes synchronization in NMSI mode.

23.4.1.1 In-Line Synchronization Pattern

The transparent channel can be programmed to receive a synchronization pattern. This pattern is defined in the data synchronization register, DSR; see Section 19.1.3, "Data Synchronization Register (DSR)." Pattern length is specified in GSMR_H[SYNL], as shown in Table 23-1. See also Section 19.1.1, "The General SCC Mode Registers (GSMR1–GSMR4)."

Table 23-1. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR.															
01	4-bit															
10	8-bit															
11	16-bit															

If a 4-bit SYNC is selected, reception begins as soon as these four bits are received, beginning with the first bit following the 4-bit SYNC. The transmitter synchronizes on the receiver pattern if GSMR_H[RSYN] = 1.

NOTE

The transparent controller does not automatically send the synchronization pattern; therefore, the synchronization pattern must be included in the transmit buffer.

23.4.1.2 External Synchronization Signals

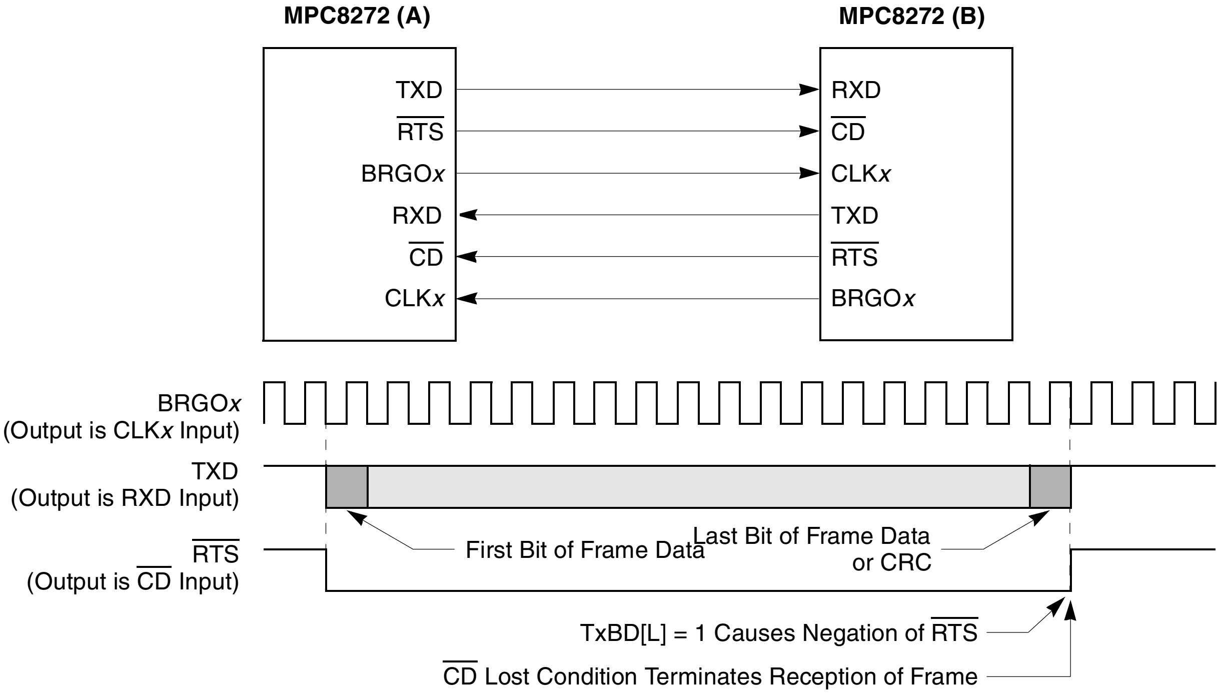
If GSMR_H[SYNL] is 0b00, the transmitter uses $\overline{\text{CTS}}$ and the receiver uses $\overline{\text{CD}}$ to begin the sequence. These signals share two options—pulsing and sampling.

GSMR_H[CDP] and GSMR_H[CTSP] determine whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must remain asserted for the duration of the transparent frame. Pulse operation allows an uninterrupted stream of data. However, use envelope mode to identify frames of transparent data.

The sampling option determines the delay between CD and CTS being asserted and the resulting action by the SCC. Assume either that these signals are asynchronous to the data and internally synchronized by the SCC or that they are synchronous to the data with faster operation. This option allows RTS of one SCC to be connected to CD of another SCC and to have the data synchronized and bit aligned. Another option is to link the transmitter synchronization to the receiver synchronization. Diagrams for the pulse/envelope and sampling options are shown in Section 23.4, “Achieving Synchronization in Transparent Mode.”

23.4.1.2.1 External Synchronization Example

Figure 23-1 shows synchronization using external signals.



Notes:

1. Each MPC8272 generates its own transmit clocks. If the transmit and receive clocks are the same, one MPC8272 can generate transmit and receive clocks for the other MPC8272. For example, CLK*x* on MPC8272 (B) could be used to clock the transmitter and receiver.
2. CTS should be configured as always asserted in the parallel I/O or connected to ground externally.
3. The required GSMR configurations are DIAG= 00, CTSS=1, CTSP is a “do not care”, CDS=1, CDP=0, TTX=1, and TRX=1. REVD and TCRC are application-dependent.
4. The transparent frame contains a CRC if TxBD[TC] is set.

Figure 23-1. Sending Transparent Frames between MPC8272s

MPC8272(A) and MPC8272(B) exchange transparent frames and synchronize each other using RTS and CD. However, CTS is not required because transmission begins at any time. Thus, RTS is connected directly to the other MPC8272 CD pin. GSMR_H[RSYN] is not used and transmission and reception from each MPC8272 are independent.

23.4.1.3 Transparent Mode without Explicit Synchronization

If there is no need to synchronize the transparent controller at a specific point, the user can ‘fake’ synchronization in one of the following ways:

- Tie a parallel I/O pin to the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ lines. Then, after enabling the receiver and transmitter, provide a falling edge by manipulating the I/O pin in software.
- Enable the receiver and transmitter for the SCC in loopback mode and then change GSMR_L[DIAG] to 0b00 while the transmitter and receiver and enabled.

23.4.2 Synchronization and the TSA

A transparent-mode SCC using the time-slot assigner can synchronize either on a user-defined inline pattern or by inherent synchronization.

Note that when using the TSA, a newly-enabled transmitter sends from 10 to 15 frames of idles before sending the actual transparent data due to startup requirements of the TDM. Therefore, when loopback testing through the TDM, expect to receive several bytes of 0xFF before the actual data.

23.4.2.1 Inline Synchronization Pattern

The receiver can be programmed to begin receiving data into the receive buffers only after a specified data pattern arrives. To synchronize on an inline pattern:

- Set GSMR_H[SYNL].
- Program the DSR with the desired pattern.
- Clear GSMR_H[CDP].
- Set GSMR_H[CTSP, CTSS, CDS].

If GSMR_H[TXSY] is also used, the transmitter begins transmission 8 clocks after the receiver achieves synchronization.

23.4.2.2 Inherent Synchronization

Inherent synchronization assumes synchronization by default when the channel is enabled; all data sent from the TDM to the SCC is received. To implement inherent synchronization:

- Set GSMR_H[CDP, CDS, CTSP, CTSS].

If these bits are not set, the received bit stream will be bit-shifted. The SCC loses the first received bit because $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are treated as asynchronous signals.

23.4.3 End of Frame Detection

An end of frame cannot be detected in the transparent data stream since there is no defined closing flag in transparent mode. Therefore, if framing is needed, the user must use the $\overline{\text{CD}}$ line to alert the transparent controller of an end of frame.

23.5 CRC Calculation in Transparent Mode

The CRC calculations follow the ITU/IEEE standard. The CRC is calculated on the transmitted data stream; that is, from lsb to msb for non-bit-reversed (GSMR_H[REVD] = 0) and from msb to lsb for bit-reversed (GSMR_H[REVD] = 1) transmission. The appended CRC is sent msb to lsb. When receiving, the CRC is calculated as the incoming bits arrive. The optional reversal of data (GSMR_H[REVD] = 1) is done just before data is stored in memory (after the CRC calculation).

23.6 SCC Transparent Parameter RAM

For transparent mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 23-2.

Table 23-2. SCC Transparent Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x30	CRC_P	Long	CRC preset for totally transparent. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF and for the CRC-16, initialize with ones (0x0000_FFFF) or zeros (0x0000_0000).
0x34	CRC_C	Long	CRC constant for totally transparent receiver. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For the 32-bit CRC-CCITT, CRC_C initialize with 0xDEBB_20E3 and for the CRC-16, which is normally used with BISYNC, initialize with 0x0000_0000.

[1] From SCC base address. See Section 19.3.1, “SCC Base Addresses.”

CRC_P and CRC_C overlap with the CRC parameters for the HDLC-based protocols. However, this overlap is not detrimental since the CRC constant is used only for the receiver and the CRC preset is used only for the transmitter, so only one entry is required for each. Thus, the user can choose an HDLC transmitter with a transparent receiver or a transparent transmitter with an HDLC receiver.

23.7 SCC Transparent Commands

The following transmit and receive commands are issued to the CP command register. Table 23-3 describes transmit commands.

Table 23-3. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 clocks (or immediately if TODR[TOD] = 1). STOP TRANSMIT disables frame transmission on the transmit channel. If the transparent controller receives the command during frame transmission, transmission is aborted after a maximum of 64 additional bits and the transmit FIFO is flushed. The current TxBD pointer (TBPTR) is not advanced, no new BD is accessed and no new buffers are sent for this channel. The transmitter will send idles.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly, rather than abruptly, in much the same way that the regular STOP TRANSMIT command stops. It stops transmission after the current frame finishes or immediately if no frame is being sent. A transparent frame is not complete until a BD with TxBD[L] set has its buffer completely sent. SCCE[GRA] is set once transmission stops; transmit parameters and their BDs can then be modified. The current TxBD pointer (TBPTR) advances to the next TxBD in the table. Transmission resumes once TxBD[R] is set and a RESTART TRANSMIT command is issued.

Table 23-3. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Reenables transmission of characters on the transmit channel. The transparent controller expects it after a STOP TRANSMIT command is issued (at which point the channel is disabled in SCCM), after a GRACEFUL STOP TRANSMIT command is issued, or after a transmitter error. The transparent controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel parameter RAM to reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

Table 23-4 describes receive commands.

Table 23-4. Receive Commands

Command	Description
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled, the channel is in receive enable mode and uses the first BD in the table. ENTER HUNT MODE forces the transparent receiver to the current frame and enter hunt mode where the transparent controller waits for the synchronization sequence. After receiving the command, the current buffer is closed. Further data reception uses the next BD.
CLOSE RXBD	Forces the SCC to close the RxBD if it is being used and to use the next BD for any subsequently received data. If the SCC is not receiving data, no action is taken by this command.
INIT RX PARAMETERS	Initializes all receive parameters in this serial channel parameter RAM to reset state. Issue only when the receiver is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

23.8 Handling Errors in the Transparent Controller

The SCC reports message reception and transmission errors using the channel buffer descriptors, the error counters, and SCCE. Table 23-5 describes transmit errors.

Table 23-5. Transmit Errors

Error	Description
Transmitter underrun	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[UN], and generates a TXE interrupt if it is enabled. Transmission resumes after a RESTART TRANSMIT command is received. Underrun occurs after a transmit frame for which TxBD[L] was not set. In this case, only SCCE[TXE] is set. Underrun cannot occur between transparent frames.
$\overline{\text{CTS}}$ lost during message transmission	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[CT], and generates the TXE interrupt if it is enabled. The channel resumes sending after RESTART TRANSMIT is received.

Table 23-6 describes receive errors.

Table 23-6. Receive Errors

Error	Description
Overrun	The SCC maintains a receive FIFO. The CPM starts programming the SDMA channel if the buffer is in external memory and updating the CRC when 8 or 32 bits are received in the FIFO as determined by GSMR_H[RFW]. If a FIFO overrun occurs, the SCC writes the received byte over the previously received byte. The previous character and its status bits are lost. Afterwards, the channel closes the buffer, sets OV in the BD, and generates the RXB interrupt if it is enabled. The receiver immediately enters hunt mode.
$\overline{\text{CD}}$ lost during message reception	When this occurs, the channel stops receiving messages, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if it is enabled. This error has highest priority. The rest of the message is lost, and no other errors are checked in the message. The receiver immediately enters hunt mode.

23.9 Transparent Mode and the PSMR

The protocol-specific mode register (PSMR) is not used by the transparent controller because all transparent mode selections are made in the GSMR. If only half of an SCC (transmitter or receiver) is running the transparent protocol, the other half (receiver or transmitter) can support another protocol. In such a case, use the PSMR for the non-transparent protocol.

23.10 SCC Transparent Receive Buffer Descriptor (RxBD)

The CPM reports information about the received data for each buffer using an RxBD, closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- An error is detected.
- A full receive buffer is detected.
- An ENTER HUNT MODE command is issued.
- A CLOSE RXBD command is issued.

Figure 23-2 displays an SCC transparent 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**	—	DE	—		NO	—	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 23-2. SCC Transparent Receive Buffer Descriptor (RxBD)

Table 23-7 describes RxBD status and control fields.

Table 23-7. SCC Transparent RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or stopped receiving data because an error occurred. The core can read or write to any fields of this RxBD. The CPM does not use this BD when RxBD[E] is zero. 1 The buffer is not full. This RxBD and buffer are owned by the CPM. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives data into the first BD that RBASE points to. The number of BDs in this table is determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is used. 1 When this buffer is closed by the transparent controller, the SCCE[RXB] is set. SCCE[RXB] can cause an interrupt if it is enabled.
4	L	Last in frame. Set by the transparent controller when this buffer is the last in a frame, which occurs when $\overline{\text{CD}}$ is negated (if GSMR_H[CDP] = 0) or an error is received. If an error is received, one or more of RxBD[OV, CD, DE] are set. Note that the SCC transparent controller writes the number of buffer (not frame) octets to the last BD's data length field. 0 Not the last buffer in a frame 1 Last buffer in a frame
5	F	First in frame. The transparent controller sets F when this buffer is the first in the frame: 0 Not the first buffer in a frame 1 First buffer in a frame
6	**CM**	Continuous mode. 0 Normal operation 1 The CPM does not clear RxBD[E] after this BD is closed, letting the buffer be overwritten when the CPM next accesses this BD. However, RxBD[E] is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared
8	DE	DPLL error. Set by the transparent controller when a DPLL error occurs as this buffer is received. In decoding modes, where a transition is promised every bit, DE is set when a missing transition occurs. If a DPLL error occurs, no other error checking is performed.
9–10	—	Reserved, should be cleared
11	NO	Rx non-octet. Set when a frame containing a number of bits not exactly divisible by eight is received.
12	—	Reserved, should be cleared
13	CR	CRC error indication bits. Indicates that this frame contains a CRC error. The received CRC bytes are always written to the receive buffer. CRC checking cannot be disabled, but it can be ignored.
14	OV	Overrun. Indicates that a receiver overrun occurred during buffer reception.
15	CD	Carrier detect lost. Indicates when $\overline{\text{CD}}$ is negated during buffer reception.

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).” The Rx buffer pointer must be divisible by four, unless GSMR_H[RFW] is set to 8 bits wide, in which case the pointer can be even or odd. The buffer can reside in internal or external memory.

23.11 SCC Transparent Transmit Buffer Descriptor (TxBD)

Data is sent to the CPM for transmission on an SCC channel by arranging it in buffers referenced by the TxBD table. The CPM uses BDs to confirm transmission or indicate error conditions so the processor knows buffers have been serviced. Prepare status and control bits before transmission; they are set by the CPM after the buffer is sent.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	—							UN	CT
Offset + 2	**Data Length**															
Offset + 4	**Tx Buffer Pointer**															
Offset + 6																

Figure 23-3. SCC Transparent Transmit Buffer Descriptor (TxBD)

Table 23-8 describes SCC transparent TxBD status and control fields.

Table 23-8. SCC Transparent TxBD Status and Control Field Descriptions

Bit	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. The BD and buffer can be updated. The CPM clears R after the buffer is sent or after an error is encountered. 1 The user-prepared buffer is not sent yet or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is determined only by TxBD[W] and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. Note that clearing this bit does not disable all SCCE[TXE] events. 0 No interrupt is generated after this buffer is serviced; SCCE[TXE] is unaffected. 1 When the CPM services this buffer, SCCE[TXB] or SCCE[TXE] is set. These bits can cause interrupts if they are enabled.
4	**L**	Last in message. 0 The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer is sent immediately after the last byte of this buffer. 1 The last byte in the buffer is the last byte in the transmitted transparent frame. After this buffer is sent, the transmitter requires synchronization before the next buffer is sent.
5	**TC**	Transmit CRC. 0 No CRC sequence is sent after this buffer. 1 A frame check sequence defined by GSMR_H[TCRC] is sent after the last byte of this buffer.

Table 23-8. SCC Transparent TxBD Status and Control Field Descriptions (continued)

Bit	Name	Description
6	**CM**	Continuous mode. 0 Normal operation 1 The CPM does not clear TxBD[R] after this BD is closed, so the buffer is automatically resent when the CPM accesses this BD next. However, TxBD[R] is cleared if an error occurs during transmission, regardless of how CM is set.
7–13	—	Reserved, should be cleared
14	UN	Underrun. Set when the SCC encounters a transmitter underrun condition while sending the buffer.
15	CT	CTS lost. Indicates the $\overline{\text{CTS}}$ was lost during frame transmission.

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).” Although it is never modified by the CP, data length should be greater than zero. The buffer pointer can be even or odd and can reside in internal or external memory.

23.12 SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

When the SCC is in transparent mode, the SCC event register (SCCE) functions as the transparent event register to report events recognized by the transparent channel and to generate interrupts. When an event is recognized, the transparent controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the transparent mask register (SCCM).

Event bits are reset by writing ones; writing zeros has no effect. All unmasked bits must be reset before the CP clears the internal interrupt request to the SIU interrupt controller. Figure 23-4 shows the event and mask registers.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—					DCC	—		GRA	—		TXE	—	BSY	TXB	RXB
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11A10 (SCCE1); 0x11A50 (SCCE3); 0x11A70 (SCCE4) 0x11A14 (SCCM1); 0x11A54 (SCCM3); 0x11A74 (SCCM4)															

Figure 23-4. SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

Table 23-9 describes SCCE/SCCM fields.

Table 23-9. SCCE/SCCM Field Descriptions [1]

Bit	Name	Description
0–4	—	Reserved, should be cleared. Refer to note 1 below.
5	DCC	DPLL CS changed. Set when the DPLL-generated carrier sense status changes (valid only when the DPLL is used). Real-time status can be read in SCCS. This is not the $\overline{\text{CD}}$ status mentioned elsewhere.
6–7	—	Reserved, should be cleared. Refer to note 1 below.

Table 23-9. SCCE/SCCM Field Descriptions (continued)[1]

Bit	Name	Description
8	GRA	Graceful stop complete. Set when a graceful stop initiated by completes as soon as the transmitter finishes any frame in progress when the GRACEFUL STOP TRANSMIT command was issued. Immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared. Refer to note 1 below.
11	TXE	Tx error. Set when an error occurs on the transmitter channel. This event is not maskable via the TxBD[I] bit.
12	—	Reserved, should be cleared. Refer to note 1 below.
13	BSY	Busy condition. Set when a byte or word is received and discarded due to a lack of buffers. The receiver resumes reception after it gets an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set no sooner than when the last bit of the last byte of the buffer begins transmission, assuming L is set in the TxBD. If it is not, TXB is set when the last byte is written to the transmit FIFO.
15	RXB	Rx buffer. Set when a complete buffer was received on the SCC channel, no sooner than two serial clocks after the last bit of the last byte in which the buffer is received on RXD.

[1] Reserved bits in the SCCE should not be masked in the SCCM register.

23.13 SCC Status Register in Transparent Mode (SCCS)

The SCC status register (SCCS) allows monitoring of real-time status conditions on the RXD line. The real-time status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are part of the parallel I/O.

	0	1	2	3	4	5	6	7
Field	—						CS	—
Reset	0000_0000							
R/W	R							
Addr	0x11A17 (SCCS1); 0x11A57 (SCCS3); 0x11A77 (SCCS4)							

Figure 23-5. SCC Status Register in Transparent Mode (SCCS)

Table 23-10 describes SCCS fields.

Table 23-10. SCCS Field Descriptions

Bit	Name	Description
0–5	—	Reserved, should be cleared
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared

23.14 SCC4 Transparent Programming Example

The following initialization sequence enables the transmitter and receiver, which operate independently of each other. They implement the connection shown on MPC8272(B) in Figure 23-1.

The transmit and receive clocks are externally provided to MPC8272(B) using CLK5. SCC4 is used. The transparent controller is configured with the $\overline{\text{RTS4}}$ and $\overline{\text{CD4}}$ pins active and $\overline{\text{CTS4}}$ is configured to be grounded internally. A 16-bit CRC-CCITT is sent with each transparent frame. The FIFOs are configured for fast operation.

1. Configure port D pins to enable TXD4 and RXD4. Set PPARD[21,22] and PDIRD[21] and clear PDIRD[22] and PSORD[21,22].
2. Configure ports C and D pins to enable $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$ and $\overline{\text{CD4}}$. Set PPARC[8,9], PPARD[20] and PDIRD[20] and clear PDIRC[8,9], PSORC[8,9] and PSORD[20].
3. Configure port C pin 27 to enable the CLK5 pin. Set PPARC[27] and clear PDIRC[27] and PSORC[27].
4. Connect CLK5 to SCC4 using the CPM mux. Program CMXSCR[R4CS] and CMXSCR[T4CS] to 0b100.
5. Connect the SCC4 to the NMSI and clear CMXSCR[SC4].
6. Write RBASE with 0x0000 and TBASE with 0x0008 in the SCC4 parameter RAM to point to one RxBD at the beginning of dual-port RAM followed by one TxBD.
7. Write 0x0CE1_0000 to the CPCR to execute INIT RX AND TX PARAMETERS for SCC4.
8. Write 0x0041 to the CPCR to execute INIT RX AND TX PARAMETERS for SCC4.
9. Write RFCR and TFCR with 0x10 for normal operation.
10. Write MRBLR with the maximum number of bytes per receive buffer and assume 16-bytes, so MRBLR = 0x0010.
11. Write CRC_P with 0x0000_FFFF to comply with the 16-bit CRC-CCITT.
12. Write CRC_C with 0x0000_F0B8 to comply with the 16-bit CRC-CCITT.
13. 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].
14. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xBC00 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
15. Write 0xFFFF to SCCE to clear any previous events.
16. Write 0x0013 to SCCM to enable the TXE, TXB, and RXB interrupts.
17. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so SCC4 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
18. Write 0x0000_1980 to GSMR_H4 to configure the transparent channel.
19. Write 0x0000_0000 to GSMR_L4 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits). Normal operation of the transmit clock is used. Note that the transmitter (ENT) and receiver (ENR) are not enabled yet.

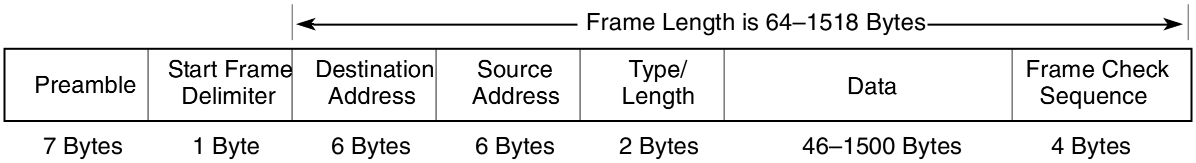
20. Write 0x0000_0030 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits are enabled last.

NOTE

After 5 bytes are sent, the Tx buffer is closed and after 16 bytes are received the Rx buffer is closed. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

Chapter 24
SCC Ethernet Mode

The Ethernet IEEE 802.3 protocol is a widely used LAN protocol based on the carrier sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. Figure 24-1 shows Ethernet and IEEE 802.3 frame structure.



NOTE: The lsb of each octet is transmitted first.

Figure 24-1. Ethernet Frame Structure

The frame begins with a 7-byte preamble of alternating ones and zeros. Because the frame is Manchester encoded, the preamble gives receiving stations a known pattern on which to lock. The start frame delimiter follows the preamble, signifying the beginning of the frame. The 48-bit destination address is next, followed by the 48-bit source address. Original versions of the IEEE 802.3 specification allowed 16-bit addressing, but this addressing has never been widely used and is not supported.

The next field is the Ethernet type field/IEEE 802.3 length field. The type field signifies the protocol used in the rest of the frame and the length field specifies the length of the data portion of the frame. For Ethernet and IEEE 802.3 frames to coexist on the same LAN, the length field of the frame must always be different from any type fields used in Ethernet. This limits the length of the data portion of the frame to 1,500 bytes and total frame length to 1,518 bytes. The last 4 bytes of the frame are the frame check sequence (FCS), a standard 32-bit CCITT-CRC polynomial used in many protocols.

When a station needs to transmit, it checks for LAN activity. When the LAN is silent for a specified period, the station starts sending. At that time, the station continually checks for collisions on the LAN; if one is found, the station forces a jam pattern (all ones) on its frame and stops sending. Most collisions occur close to the beginning of a frame. The station waits a random period of time, called a backoff, before trying to retransmit. Once the backoff time expires, the station waits for silence on the LAN before retransmitting, which is called a retry. If the frame cannot be sent within 15 retries, an error occurs

10-Mbps Ethernet transmits at 0.8 µs per byte. The preamble plus start frame delimiter is sent in 6.4 µs. The minimum 10-Mbps Ethernet interframe gap is 9.6 µs, and the slot time is 52 µs.

24.1 Ethernet on the MPC8272

Setting GSMR[MODE] to 0b1100 selects Ethernet. The SCC performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions.

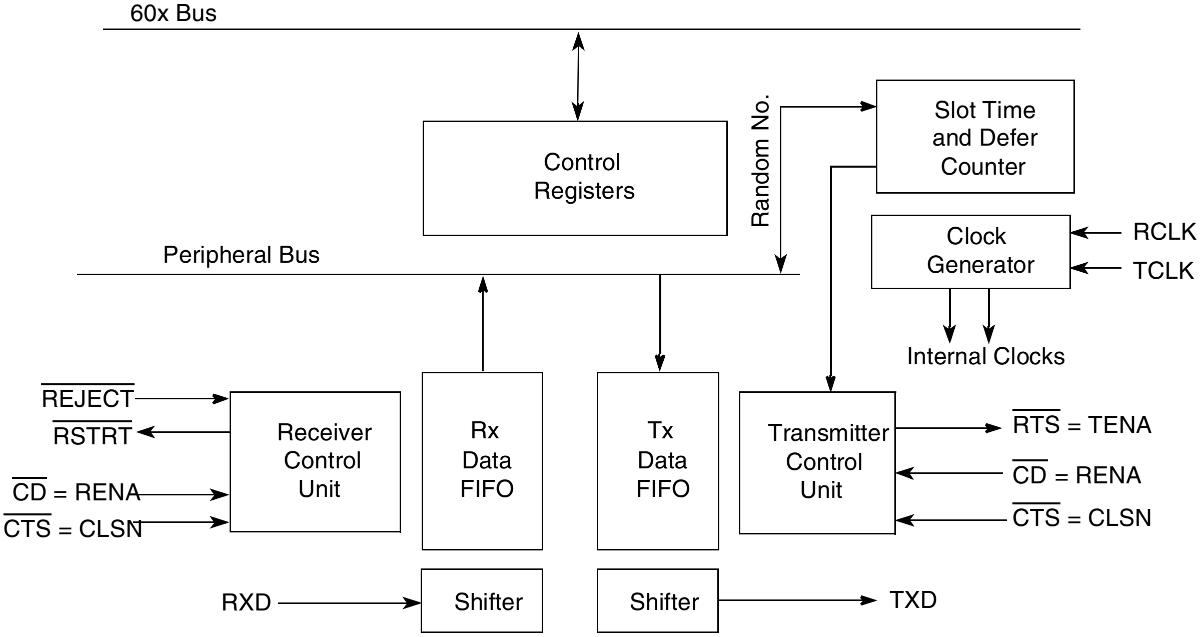


Figure 24-2. Ethernet Block Diagram

The MPC8272 Ethernet controller requires an external serial interface adaptor (SIA) and transceiver function to complete the interface to the media.Although the MPC8272 contains DPLLs that allow Manchester encoding and decoding, these DPLLs were not designed for Ethernet rates. Therefore, the MPC8272 Ethernet controller bypasses the on-chip DPLLs and uses the external system interface adaptor on the EEST instead. The on-chip DPLL cannot be used for low-speed (1-Mbps) Ethernet either because it cannot properly detect start-of-frame or end-of-frame.

Note that the CPM of the MPC8272 requires a minimum system clock frequency of 24 MHz to support Ethernet.

24.2 Features

The following list summarizes the main features of the SCC in Ethernet mode:

- Performs MAC layer functions of Ethernet and IEEE 802.3
- Performs framing functions
  - Preamble generation and stripping
  - Destination address checking
  - CRC generation and checking
  - Automatically pads short frames on transmit
  - Framing error (dribbling bits) handling
- Full collision support
  - Enforces the collision (jamming)
  - Truncated binary exponential backoff algorithm for random wait
  - Two nonaggressive backoff modes

  - Automatic frame retransmission (until the attempt limit is reached)
  - Automatic discard of incoming collided frames
  - Delay transmission of new frames for specified interframe gap
- Maximum 10-Mbps bit rate
- Optional full-duplex support
- Back-to-back frame reception
- Detection of receive frames that are too long
- Multibuffer data structure
- Supports 48-bit addresses in three modes
  - Physical: One 48-bit address recognized or 64-bin hash table for physical addresses
  - Logical: 64-bin group address hash table plus broadcast address checking
  - Promiscuous: Receives all addresses, but discards frame if $\overline{\text{REJECT}}$ is asserted
- External content-addressable memory (CAM) support on serial bus interfaces
- Up to eight parallel I/O pins can be sampled and appended to any frame
- Optional heartbeat indication
- Transmitter network management and diagnostics
  - Lost carrier sense
  - Underrun
  - Number of collisions exceeded the maximum allowed
  - Number of retries per frame
  - Deferred frame indication
  - Late collision
- Receiver network management and diagnostics
  - CRC error indication
  - Nonoctet alignment error
  - Frame too short
  - Frame too long
  - Overrun
  - Busy (out of buffers)
- Error counters
  - Discarded frames (out of buffers or overrun occurred)
  - CRC errors
  - Alignment errors
- Internal and external loopback mode

24.3 Connecting the MPC8272 to Ethernet

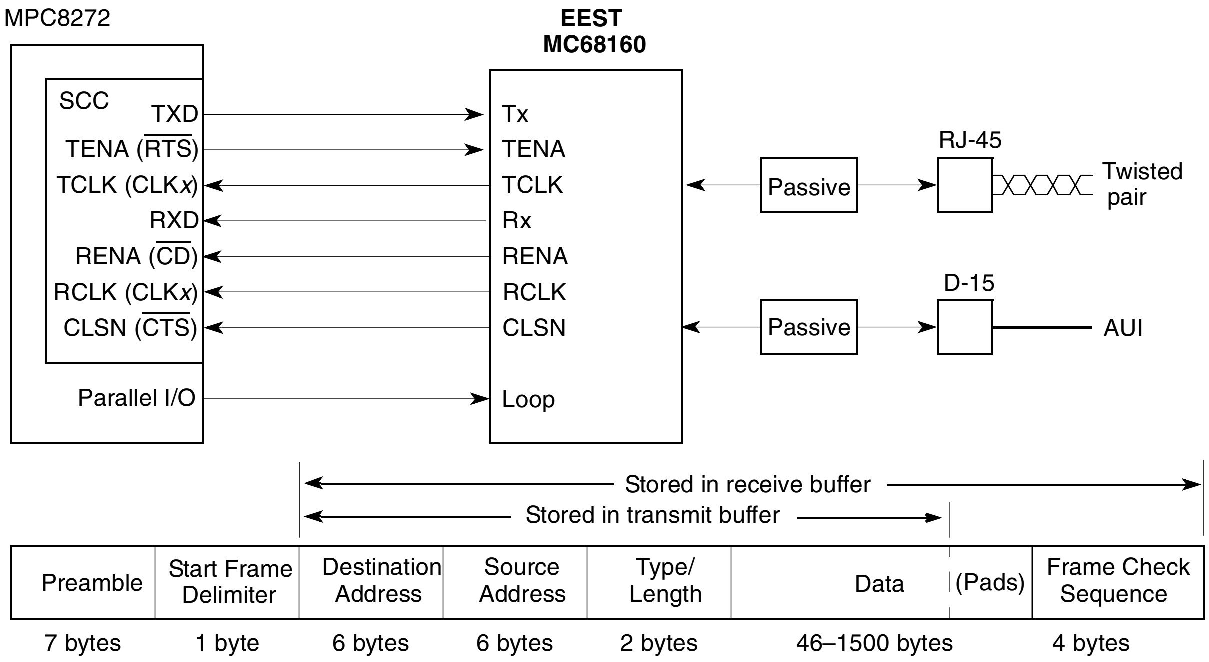
The basic interface to the external SIA chip consists of the following Ethernet signals:

- Receive clock (RCLK)—A CLK*x* signal routed through the bank of clocks on the MPC8272
- Transmit clock (TCLK)—A CLK*x* signal routed through the bank of clocks on the MPC8272. Note that RCLK and TCLK should not be connected to the same CLK*x* since the SIA provides separate transmit and receive clock signals.
- Transmit data (TXD)—The MPC8272 TXD signal
- Receive data (RXD)—The MPC8272 RXD signal

The following signals take on different functionality when the SCC is in Ethernet mode:

- Transmit enable (TENA)—$\overline{\text{RTS}}$ becomes TENA. The polarity of TENA is active high, whereas the polarity of $\overline{\text{RTS}}$ is active low.
- Receive enable (RENA)—$\overline{\text{CD}}$ becomes RENA.
- Collision (CLSN)—$\overline{\text{CTS}}$ becomes CLSN. The carrier sense signal is referenced in Ethernet descriptions because it indicates when the LAN is in use. Carrier sense is defined as the logical OR of RENA and CLSN.

Figure 24-3 shows the basic components and signals required to make an Ethernet connection between the MPC8272 and EEST.



NOTE: Short Tx frames are padded automatically by the MPC8272.

Figure 24-3. Connecting the MPC8272 to Ethernet

The EEST has similar names for its connection to the above seven MPC8272 signals. The EEST also provides a loopback input so the MPC8272 can perform external loopback testing, which can be controlled by any available MPC8272 parallel I/O signal. The passive components needed to connect to AUI or twisted-pair media are external to the EEST. The MC68160 documentation describes EEST connection circuits.

The MPC8272 uses SDMA channels to store bytes received after the start frame delimiter in system memory. When sending, provide the destination address, source address, type/length field, and the transmit data. To meet minimum frame requirements, the MPC8272 pads frames with fewer than 46 bytes in the data field and appends the FCS to the frame.

24.4 SCC Ethernet Channel Frame Transmission

The Ethernet transmitter works with almost no core intervention. When the core enables the transmitter, the SCC polls the first TxBD in the table every 128 serial clocks. Setting TODR[TOD] lets the next frame be sent without waiting for the next poll.

To begin transmission, the SCC in Ethernet mode (called the Ethernet controller) fetches data from the buffer, asserts TENA to the EEST, and starts sending the preamble sequence, the start frame delimiter, and frame information. If the line is busy, it waits for carrier sense to remain inactive for 6.0 µs, at which point it waits an additional 3.6 µs before it starts sending (9.6 µs after carrier sense originally became inactive).

If a collision occurs during frame transmission, the Ethernet controller follows a specified backoff procedure and tries to retransmit the frame until the retry limit threshold is reached. The Ethernet controller stores the first 5 to 8 bytes of the transmit frame in dual-port RAM so they need not be retrieved from system memory in case of a collision. This improves bus usage and latency when the backoff timer output requires an immediate retransmission. If a collision occurs during frame transmission, the controller returns to the first buffer for a retransmission. The only restriction is that the first buffer must contain at least 9 bytes.

NOTE

If an Ethernet frame consists of multiple buffers, do not reuse the first BD until the CPM clears the R bit of the last BD.

When the end of the current BD is reached and TxBD[L] is set, the FCS bytes are appended (if the TC bit is set in the TxBD), and TENA is negated. This notifies the EEST of the need to generate the illegal Manchester encoding that marks the end of an Ethernet frame. After CRC transmission, the Ethernet 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 bit is not set, only the R bit is cleared.

In either mode, whether an interrupt is issued depends on how the I bit is set in the TxBD. The Ethernet controller proceeds to the next TxBD. Transmission can be interrupted after each frame, after each buffer, or after a specific buffer is sent. The Ethernet controller can pad characters to short frames. If TxBD[PAD] is set, the frame is padded up to the value of the minimum frame length register (MINFLR).

To send expedited data before previously linked buffers or for error situations, the GRACEFUL STOP TRANSMIT command can be used to rearrange transmit queue before the CPM sends all the frames; the Ethernet controller stops immediately if no transmission is in progress or it will keep sending until the current frame either finishes or terminates with a collision. When the Ethernet controller receives a RESTART TRANSMIT command, it resumes transmission. The Ethernet controller sends bytes least-significant bit first.

24.5 SCC Ethernet Channel Frame Reception

The Ethernet receiver handles address recognition and performs CRC, short frame, maximum DMA transfer, and maximum frame length checking with almost no core intervention. When the core enables the Ethernet receiver, it enters hunt mode as soon as RENA is asserted while CLSN is negated. In hunt mode, as data is shifted into the receive shift register one bit at a time, the register contents are compared to the contents of the SYN1 field in the data synchronization register (DSR). This compare function becomes valid a certain number of clocks after the start of the frame (depending on PSMR[NIB]). If the two are not equal, the next bit is shifted in and the comparison is repeated. If a double-zero or double-one fault is detected between bits 14 and 21 from the first received preamble bit, the frame is rejected. If a double-zero fault is detected after 21 bits from the first received preamble bit and before detection of the start frame delimiter (SFD), the frame is also rejected. When the incoming pattern is not rejected and matches the DSR, the SFD has been detected; hunt mode is terminated and character assembly begins.

When the receiver detects the first bytes of the frame, the Ethernet controller performs address recognition on the frame. The receiver can receive physical (individual), group (multicast), and broadcast addresses. Ethernet receive frame data is not written to memory until the internal address recognition process completes, which improves bus usage with frames not addressed to this station.

If a match is found, the Ethernet controller fetches the next RxBD and, if it is empty, starts transferring the incoming frame to the RxBD associated data buffer. If a collision is detected during the frame, the RxBDs associated with this frame are reused. Thus, there will be no collision frames presented to you except late collisions, which indicate serious LAN problems. When the data buffer has been filled, the Ethernet controller clears the E bit in the RxBD and generates an interrupt if the I bit is set. If the incoming frame exceeds the length of the data buffer, the Ethernet controller fetches the next RxBD in the table and, if it is empty, continues transferring the rest of the frame to this buffer. The RxBD length is determined by MRBLR in the SCC general-purpose parameter RAM, which should be at least 64 bytes.

During reception, the Ethernet controller checks for a frame that is either too short or too long. When the frame ends, the receive CRC field is checked and written to the buffer. The data length written to the last BD in the Ethernet frame is the length of the entire frame and it enables the software to correctly recognize the frame-too-long condition.

The Ethernet controller then sets the L bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E bit. Then it generates a maskable interrupt, which indicates that a frame has been received and is in memory. The Ethernet controller then waits for a new frame. It receives serial data least-significant bit first.

24.6 The Content-Addressable Memory (CAM) Interface

The Ethernet controller has one option for connecting to an external CAM—a serial interface. The reject signal ($\overline{\text{REJECT}}$) is used to signify that the current frame should be discarded. The MPC8272's internal address recognition logic can be used in combination with an external CAM. See Section 24.10, "SCC Ethernet Address Recognition."

The MPC8272 outputs a receive start ($\overline{\text{RSTRT}}$) signal when the start frame delimiter is recognized. This signal is asserted for one bit time on the second destination address bit. The CAM control logic uses $\overline{\text{RSTRT}}$ (in combination with the RXD and RCLK signals) to store the destination or source address and generate writes to the CAM for address recognition. In addition, the RENA signal supplied from the SIA can be used to abort the comparison if a collision occurs on the receive frame.

After the comparison, the CAM control logic asserts the receive reject signal ($\overline{\text{REJECT}}$), if the current receive frame is rejected. The MPC8272’s Ethernet controller then immediately stops writing data to system memory and reuses the buffer(s) for the next frame. If the CAM accepts the frame, the CAM control logic does nothing ($\overline{\text{REJECT}}$ is not asserted). However, if $\overline{\text{REJECT}}$ is asserted, it must be done prior to the end of the receive frame.

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

24.7 SCC Ethernet Parameter RAM

For Ethernet mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 24-1.

Table 24-1. SCC Ethernet Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	**C_PRES**	Word	Preset CRC. For the 32-bit CRC-CCITT, initialize to 0xFFFF_FFFF.
0x34	**C_MASK**	Word	Constant mask for CRC. For the 32-bit CRC-CCITT, initialized to 0xDEBB_20E3.
0x38	**CRCEC**	Word	CRC error, alignment error, and discard frame counters. The CPM maintains these 32-bit (modulo 2^{32}) counters that can be initialized while the channel is disabled. CRCEC is incremented for each received frame with a CRC error, not including frames not addressed to the controller, frames received in the out-of-buffers condition, frames with overrun errors, or frames with alignment errors. ALEC is incremented for frames received with dribbling bits, but does not include frames not addressed to the controller, frames received in the out-of-buffers condition, or frames with overrun errors. DISFC is incremented for frames discarded because of the out-of-buffers condition or an overrun error. The CRC does not have to be correct for DISFC to be incremented.
0x3C	**ALEC**		
0x40	**DISFC**		
0x44	**PADS**	Hword	Short frame PAD character. Write the pad character pattern to be sent when short frame padding is implemented into PADS. The pattern may be of any value, but both the high and low bytes should be the same.
0x46	**RET_LIM**	Hword	Retry limit. Number of retries (typically 15 decimal) that can be made to send a frame. An interrupt can be generated if the limit is reached.
0x48	RET_CNT	Hword	Retry limit counter. Temporary down-counter for counting retries.
0x4A	**MFLR**	Hword	Maximum frame length register (Typically 1518 decimal). The Ethernet controller checks the length of an incoming Ethernet frame against this limit. If it is exceeded, the rest of the frame is discarded and LG is set in the last BD of that frame. The controller reports frame status and length in the last BD. MFLR is defined as all in-frame bytes between the start frame delimiter and the end of the frame.

Table 24-1. SCC Ethernet Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x4C	**MINFLR**	Hword	Minimum frame length register. The Ethernet controller checks the incoming frame's length against MINFLR (typically 64 decimal). If the received frame is smaller than MINFLR, it is discarded unless PSMR[RSH] is set, in which case, SH is set in the last BD for the frame. For transmitting a frame that is too short, the Ethernet controller pads the frame to make it MINFLR bytes long, depending on how PAD is set in the TxBD and on the PAD value in the parameter RAM.
0x4E	**MAXD1**	Hword	Max DMA*n* length register. Gives the option to stop system bus writes after a frame exceeds a certain size. However, this value is valid only if an address match is found. The Ethernet controller checks the length of an incoming Ethernet frame against this user-defined value (usually 1520 decimal). If this limit is exceeded, the rest of the incoming frame is discarded. The Ethernet controller waits until the end of the frame or until MFLR bytes are received and reports the frame status and the frame length in the last RxBD. MAXD1 is used when an address matches an individual or group address. MAXD2 is used in promiscuous mode when no address match is detected. In a monitor station, MAXD2 can be much less than MAXD1 to receive entire frames addressed to this station, but only the headers of the other frames are received.
0x50	**MAXD2**	Hword	
0x52	MAXD	Hword	Rx max DMA
0x54	DMA_CNT	Hword	Rx DMA counter. A temporary down-counter used to track frame length.
0x56	MAX_B	Hword	Maximum BD byte count
0x58	**GADDR1**	Hword	Group address filter 1–4. Used in the hash table function of the group addressing mode. Write zeros to these values after reset and before the Ethernet channel is enabled to disable all group hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table.
0x5A	**GADDR2**		
0x5C	**GADDR3**		
0x5E	**GADDR4**		
0x60	TBUF0_DATA0	Word	Save area 0—current frame.
0x64	TBUF0_DATA1	Word	Save area 1—current frame.
0x68	TBUF0_RBA0	Word	
0x6C	TBUF0_CRC	Word	
0x70	TBUF0_BCNT	Hword	
0x72	**PADDR1_H**	Hword	The 48-bit individual address of this station into this location. PADDR1_L is the lowest order hword and PADDR1_H is the highest order hword.
0x74	**PADDR1_M**		
0x76	**PADDR1_L**		
0x78	**P_PER**	Hword	Persistence. Lets the Ethernet controller be less aggressive after a collision. Normally, 0x0000. It can be a value between 1 and 9 (1 is most aggressive). The value is added to the retry count in the backoff algorithm to reduce the chance of transmission on the next time slot. Note: Using P_PER is fully allowed in the Ethernet/802.3 specifications. A less aggressive backoff algorithm used by multiple stations on a congested Ethernet LAN increases overall throughput by reducing the chance of collision. PSMR[SBT] offers another way to reduce the aggressiveness of the Ethernet controller.
0x7A	RFBD_PTR	Hword	Rx first BD pointer

Table 24-1. SCC Ethernet Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x7C	TFBD_PTR	Hword	Tx first BD pointer
0x7E	TLBD_PTR	Hword	Tx last BD pointer
0x80	TBUF1_DATA0	Word	Save area 0—next frame.
0x84	TBUF1_DATA1	Word	Save area 1—next frame.
0x88	TBUF1_RBA0	Word	
0x8C	TBUF1_CRC	Word	
0x90	TBUF1_BCNT	Hword	
0x92	TX_LEN	Hword	Tx frame length counter
0x94	**IADDR1**	Hword	Individual address filter 1–4. Used in the hash table function of the individual addressing mode. Zeros can be written to these values after reset and before the Ethernet channel is enabled to disable all individual hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table.
0x96	**IADDR2**		
0x98	**IADDR3**		
0x9A	**IADDR4**		
0x9C	BOFF_CNT	Hword	Backoff counter
0x9E	**TADDR_H**	Hword	Allows addition and deletion of addresses from individual and group hash tables. After placing an address in TADDR, issue a SET GROUP ADDRESS command. TADDR_L (temp address low) is the least-significant half word and TADDR_H (temp address high) is the most-significant half word.
0x A0	**TADDR_M**		
0x A2	**TADDR_L**		

[1] From SCC base address. See Section 19.3.1, “SCC Base Addresses.”

24.8 Programming the Ethernet Controller

The core configures the SCC to operate as an Ethernet controller by setting GSMR[MODE] to 0b1100. Receive and transmit errors are reported through RxBD and TxBD. Several GSMR fields must be programmed to special values for Ethernet. Set DSR[SYN1] to 0x55 and DSR[SYN2] to 0xD5. The 6 bytes of preamble programmed in the GSMR, in combination with the DSR programming, causes 8 bytes of preamble on transmit (including the 1-byte start delimiter with the value 0xD5).

24.9 SCC Ethernet Commands

Transmit and receive commands are issued to the CP command register (CPCR). Table 24-2 describes transmit commands.

Table 24-2. 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	Used to ensure that transmission stops smoothly after the current frame finishes or has a collision. SCCE[GRA] is set once transmission stops, at which point Ethernet transmit parameters and their BDs can be updated. TBPTR points to the next TxBD. Transmission begins once the R bit of the next BD is set and a RESTART TRANSMIT command is issued. Note that if GRACEFUL STOP TRANSMIT is issued and the current frame ends in a collision, TBPTR points to the start of the collided frame with the R bit still set in the BD. The frame looks as if it was never sent.
RESTART TRANSMIT	Enables transmission of characters on the transmit channel. The Ethernet controller expects it after a GRACEFUL STOP TRANSMIT command is issued or a transmitter error. The Ethernet controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes transmit parameters in this serial channel parameter RAM to reset state. Issue only when the transmitter is disabled. INIT TX and RX PARAMETERS resets both transmit and receive parameters.

Table 24-3 describes receive commands.

Table 24-3. Receive Commands

Command	Description
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled in GSMR_L, the channel is in receive enable mode and uses the first BD in the table. The receiver then enters hunt mode, waiting for an incoming frame. The ENTER HUNT MODE command is generally used to force the Ethernet receiver to stop receiving the current frame and enter hunt mode, in which the Ethernet controller continually scans the input data stream for a transition of carrier sense from inactive to active and then a preamble sequence followed by the start frame delimiter. After receiving the command, the buffer is closed and the CRC calculation is reset. The next RxBD is used to receive more frames.
CLOSE RXBD	Should not be used with the Ethernet controller
INIT RX PARAMETERS	Initializes receive parameters in this serial channel parameter RAM to their reset state. Issue it only when the receiver is disabled. INIT TX and RX PARAMETERS resets receive and transmit parameters.
SET GROUP ADDRESS	Used to set one of the 64 bits of the four individual/group address hash filter registers. The address to be added to the hash table should be written to TADDR_L, TADDR_M, and TADDR_H in the parameter RAM before executing this command. The CP uses an individual address if the I/G bit in the address stored in TADDR is 0; otherwise, it uses a group address. This command can be executed at any time, regardless of whether the Ethernet channel is enabled. To delete an address from the hash table, disable the Ethernet channel, clear the hash table registers, and execute this command for the remaining addresses. Do not simply clear the channel's associated hash table bit because the hash table may have multiple addresses mapped to the same hash table bit.

NOTE

After a CPM reset through CPCR[RST], the Ethernet transmit enable (TENA) signal defaults to its $\overline{\text{RTS}}$, active-low functionality. To prevent false TENA assertions to an external transceiver, configure TENA as an input before issuing a CPM reset. See step 3 in Section 24.21, "SCC Ethernet Programming Example."

24.10 SCC Ethernet Address Recognition

The Ethernet controller can filter 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. A flowchart for address recognition on received frames is shown in Figure 24-4.

In the physical type of address recognition, the Ethernet controller compares the destination address field of the received frame with the user-programmed physical address in PADDR1. Address recognition can be performed on multiple individual addresses using the IADDR1–4 hash table.

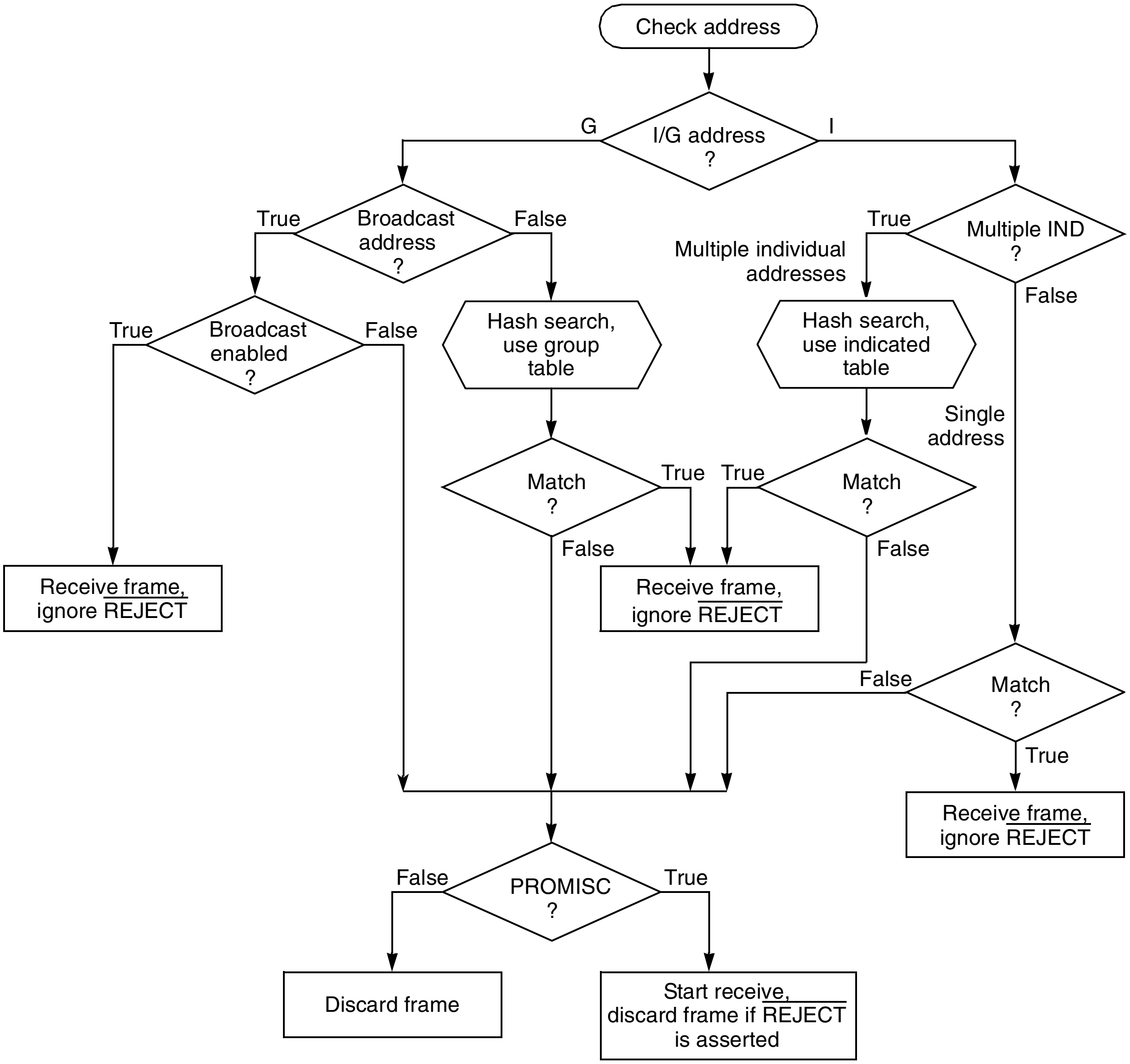

Check address
I/G address ?
G
I
Broadcast address ?
True
False
Multiple IND ?
True
False
Multiple individual addresses
Broadcast enabled ?
True
False
Hash search, use group table
Hash search, use indicated table
Single address
Match ?
True
False
Match ?
True
False
Receive frame, ignore REJECT
Receive frame, ignore REJECT
Match ?
False
True
Receive frame, ignore REJECT
PROMISC ?
False
True
Discard frame
Start receive, discard frame if REJECT is asserted

Figure 24-4. Ethernet Address Recognition Flowchart

In group address recognition, the controller determines whether the group address is a broadcast address. If broadcast addresses are enabled, the frame is accepted, but if the group address is not a broadcast address, address recognition can be performed on multiple group addresses using the GADDR*n* hash table. In promiscuous mode, the controller receives all incoming frames regardless of their address, unless $\overline{\text{REJECT}}$ is asserted.

If an external CAM is used for address recognition, select promiscuous mode; the frame can be rejected by asserting $\overline{\text{REJECT}}$ while the frame is being received. The on-chip address recognition functions can be used in addition to the external CAM address recognition functions.

If the external CAM stores addresses that should be rejected rather than accepted, the use of $\overline{\text{REJECT}}$ by the CAM should be logically inverted.

24.11 Hash Table Algorithm

Individual and group hash filtering operate using certain processes. The Ethernet controller maps any 48-bit address into one of 64 bins, each represented by a bit stored in GADDR*x* or IADDR*x*. When a SET GROUP ADDRESS command is executed, the Ethernet controller maps the selected 48-bit address into one of the 64 bits by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting 6 bits of the CRC-encoded result to generate a number between 1 and 64. Bits 31–30 of the CRC result select one of the GADDRs or IADDRs; bits 29–26 of the CRC result indicate the bit in that register.

When the Ethernet controller receives a frame, the same process is used. If the CRC generator selects a bit that is set in the group/individual hash table, the frame is accepted. Otherwise, it is rejected. So, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Frames that reach memory must be further filtered by the processor to determine if they contain one of the eight preferred addresses.

Better performance is achieved by using the group and individual hash tables simultaneously. For instance, if eight group and eight physical addresses are stored in their respective hash tables, 87.5% of all frames are prevented from reaching memory. The effectiveness of the hash table declines as the number of addresses increases. For instance, with 128 addresses stored in a 64-bin hash table, the vast majority of the hash table bits are set, thus preventing a small fraction of the frames from reaching memory.

Hash tables cannot be used to reject frames that match a set of entered addresses because unintended addresses are mapped to the same bit in the hash table.

24.12 Interpacket Gap Time

The receiver receives back-to-back frames with a minimum interpacket spacing of 9.6 µs. In addition, after the backoff algorithm, the transmitter waits for carrier sense to be negated before resending the frame. Retransmission begins 9.6 µs after carrier sense is negated if it stays negated for at least 6.4 µs.

24.13 Handling Collisions

If a collision occurs as a frame is being sent, the Ethernet controller continues sending for at least 32 bit times, thus sending a JAM pattern of 32 ones. If a collision occurs during the preamble sequence, the JAM pattern is sent at the end of the sequence.

If a collision occurs within 64 byte times, the retry process is initiated. The transmitter waits a random number of slot times (512 bit times or 52 µs). If a collision occurs after 64 byte times, no retransmission is performed and the buffer is closed with an LC error indication. If a collision occurs while a frame is being received, reception stops. This error is reported only in the BD if the length of the frame exceeds MINFLR or if PSMR[RSH] = 1.

24.14 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loop-back mode, both of the SCC FIFOs are used and the channel actually operates in a full-duplex fashion. Both internal and external loopback are configured using combinations of PSMR[LPB] and GSMR[DIAG]. Because of the full-duplex nature of the loopback operation, the performance of other SCCs is degraded.

Internal loopback disconnects the SCC from the serial interface. Receive data is connected to the transmit data and the receive clock is connected to the transmit clock. Both FIFOs are used. Data from the transmit FIFO is received immediately into the receive FIFO. There is no heartbeat check in this mode; configure TENA as a general-purpose output.

In external loopback operation, the Ethernet controller listens for data being received from the EEST at the same time that it is sending.

24.15 Full-Duplex Ethernet Support

To run full-duplex Ethernet, select loopback and full-duplex Ethernet modes in the SCC's protocol-specific mode register, (PSMR[LPB, FDE] = 1). The loopback mode tells the Ethernet controller to accept received frames without signaling a collision. Setting PSMR[FDE] tells the controller that it can send while receiving without waiting for a clear line (carrier sense).

24.16 Handling Errors in the Ethernet Controller

The Ethernet controller reports frame reception and transmission error conditions using channel BDs, error counters, and SCCE. Table 24-4 describes transmission errors.

Table 24-4. Transmission Errors

Error	Description
Transmitter underrun	If this error occurs, the channel sends 32 bits that ensures a CRC error, stops sending the buffer, closes it, sets the UN bit in the TxBD and SCCE[TXE]. The channel resumes transmission after it receives a RESTART TRANSMIT command.
Carrier sense lost during frame transmission	When this error occurs and no collision is found in the frame, the channel sets the CSL bit in the TxBD, sets SCCE[TXE], and continues sending the buffer normally. No retries are performed after this error occurs. Carrier sense is the logical OR of RENA and CLSN.
Retransmission attempts limit expired	The channel stops sending the buffer, closes it, sets the RL bit in the TxBD and SCCE[TXE]. The channel resumes transmission after it receives a RESTART TRANSMIT command.

Table 24-4. Transmission Errors (continued)

Error	Description
Late collision	When this error occurs, the channel stops sending the buffer, closes it, sets SCCE[TXE] and the LC bit in the TxBD. The channel resumes transmission after it receives the RESTART TRANSMIT command. This error is discussed further in the definition of PSMR[LCW].
Heartbeat	Some transceivers have a heartbeat (signal-quality error) self-test. To signify a good self-test, the transceiver indicates a collision to the MPC8272 within 20 clocks after the Ethernet controller sends a frame. This heartbeat condition does not imply a collision error, but that the transceiver seems to be functioning properly. If SCCE[HBC] = 1 and the MPC8272 does not detect a heartbeat condition after sending a frame, a heartbeat error occurs; the channel closes the buffer, sets the HB bit in the TxBD, and generates the TXE interrupt if it is enabled.

Table 24-5 describes reception errors.

Table 24-5. Reception Errors

Error	Description
Overrun	The Ethernet controller maintains an internal FIFO for receiving data. When it overruns, the channel writes the received byte over the previously received byte. The previous byte and frame status are lost. The channel closes the buffer, sets RxBD[OV] and SCCE[RXF], and increments the discarded frame counter (DISFC). The receiver then enters hunt mode.
Busy	A frame was received and discarded because of a lack of buffers. The channel sets SCCE[BSY] and increments DISFC. The receiver then enters hunt mode.
Non-octet error (dribbling bits)	The Ethernet controller handles up to seven dribbling bits when the receive frame terminates nonoctet aligned. It checks the CRC of the frame on the last octet boundary. If there is a CRC error, a frame nonoctet aligned error is reported, SCCE[RXF] is set, and the alignment error counter is incremented. If there is no CRC error, no error is reported. The receiver then enters hunt mode.
CRC	When a CRC error occurs, the channel closes the buffer, sets SCCE[RXF] and CR in the RxBD, and increments the CRC error counter (CRCEC). After receiving a frame with a CRC error, the receiver enters hunt mode. CRC checking cannot be disabled, but CRC errors can be ignored if checking is not required.

24.17 Ethernet Mode Register (PSMR)

In Ethernet mode, the protocol-specific mode register (PSMR), shown in Figure 24-5, is used as the Ethernet mode register.

	0	1	2	3	4	5	6	7	8	9	10	11	12		14	15
Field	HBC	FC	RSH	IAM	CRC		PRO	BRO	SBT	LPB	—	LCW	NIB			FDE
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11A08 (PSMR1); 0x11A48 (PSMR3); 0x11A68 (PSMR4)															

Figure 24-5. Ethernet Mode Register (PSMR)

Table 24-6 describes PSMR fields.

Table 24-6. PSMR Field Descriptions

Bits	Name	Description
0	HBC	Heartbeat checking. 0 No heartbeat checking is performed. Do not wait for a collision after transmission. 1 Wait 20 transmit clocks or 2 µs for a collision asserted by the transceiver after transmission. The HB bit in the TxBD is set if the heartbeat is not heard within 20 transmit clocks.
1	FC	Force collision. 0 Normal operation 1 The channel forces a collision when each frame is sent. To test collision logic configure the MPC8272 in loopback operation. In the end, the retry limit for each transmit frame is exceeded.
2	RSH	Receive short frames. 0 Discard short frames that are not as long as MINFLR. 1 Receive short frames.
3	IAM	Individual address mode. 0 Normal operation. A single 48-bit physical address in PADDR1 is checked when it is received. 1 The individual hash table is used to check all individual addresses that are received.
4–5	CRC	CRC selection. Only CRC = 10 is valid. Complies with Ethernet specifications. 32-bit CCITT-CRC. X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1.
6	PRO	Promiscuous. 0 Check the destination address of incoming frames. 1 Receive the frame regardless of its address unless $\overline{\text{REJECT}}$ is asserted as it is being received.
7	BRO	Broadcast address. 0 Receive all frames containing the broadcast address. 1 Reject all frames containing the broadcast address, unless PRO = 1.
8	SBT	Stop backoff timer. 0 The backoff timer is functioning normally. 1 The backoff timer for the random wait after a collision is stopped when carrier sense is active. Retransmission is less aggressive than the maximum allowed in IEEE 802.3. The persistence (P_PER) feature in the parameter RAM can be used in combination with or in place of SBT.
9	LPB	Local protect bit 0 Receiver is blocked when transmitter sends (default). 1 Receiver is not blocked when transmitter sends. Must be set for full-duplex operation. For loopback operation, GSMR[DIAG] must be programmed also; see Section 19.1.1, “The General SCC Mode Registers (GSMR1–GSMR4).”
10	—	Reserved. Should be cleared.
11	LCW	Late collision window. 0 A late collision is any collision that occurs at least 64 bytes from the preamble. 1 A late collision is any collision that occurs at least 56 bytes from the preamble.

Table 24-6. PSMR Field Descriptions (continued)

Bits	Name	Description
12–14	NIB	Number of ignored bits. Determines how soon after RENA assertion the Ethernet controller should begin looking for the start frame delimiter. Typically NIB = 101 (22 bits). 000 Begin searching 13 bits after the assertion of RENA. 001 Begin searching 14 bits after the assertion of RENA. ... 111 Begin searching 24 bits after the assertion of RENA.
15	FDE	Full duplex Ethernet. 0 Disable full-duplex Ethernet mode. 1 Enable full-duplex Ethernet mode. **Note:** When FDE = 1, PSMR[LPB] must also be set.

24.18 SCC Ethernet Receive BD

The Ethernet controller uses the RxBD to report on the received data for each buffer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	F	—	M	—		LG	NO	SH	CR	OV	CL
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 24-6. SCC Ethernet RxBD

Table 24-7 describes RxBD status and control fields.

Table 24-7. SCC Ethernet RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty. 0 The buffer is full or stopped receiving data because an error occurred. The core can read or write any fields of this RxBD. The CPM does not use this BD as long as the E bit is zero. 1 The buffer is not full. The CPM controls this BD and its buffer; do not modify this BD.
1	—	Reserved, should be cleared
2	W	Wrap (final BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives incoming data into the first BD that RBASE points to. The number of BDs is determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt. Note that this bit does not mask SCCE[RXF] interrupts. 0 No SCCE[RXB] interrupt is generated after this buffer is used. 1 SCCE[RXB] or SCCE[RXF] is set when this buffer is used by the Ethernet controller. These two bits can cause interrupts if they are enabled.

Table 24-7. SCC Ethernet RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
4	L	Last in frame. The Ethernet controller sets this bit when this buffer is the last one in a frame, which occurs when the end of a frame is reached or an error is received. In the case of error, one or more of the CL, OV, CR, SH, NO, and LG bits are set. The Ethernet controller writes the number of frame octets to the data length field. 0 The buffer is not the last one in a frame. 1 The buffer is the last one in a frame.
5	F	First in frame. The Ethernet controller sets this bit when this buffer is the first one in a frame. 0 The buffer is not the first one in a frame. 1 The buffer is the first one in a frame.
6	—	Reserved, should be cleared.
7	M	Miss. (valid only if L = 1) The Ethernet controller sets M for frames that are accepted in promiscuous mode, but are flagged as a miss by internal address recognition. Thus, in promiscuous mode, M determines whether a frame is destined for this station. 0 The frame is received because of an address recognition hit. 1 The frame is received because of promiscuous mode.
8–9	—	Reserved, should be cleared
10	LG	Rx frame length violation. Set when a frame length greater than the maximum defined for this channel has been recognized. Only the maximum number of bytes allowed is written to the buffer.
11	NO	Rx nonoctet-aligned frame. Set when a frame containing a number of bits not divisible by eight is received. Also, the CRC check that occurs at the preceding byte boundary generated an error.
12	SH	Short frame. Set if a frame smaller than the minimum defined for this channel was recognized. Occurs if PSMR[RSH] = 1.
13	CR	Rx CRC error. set when a frame contains a CRC error.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CL	Collision. This frame is closed because a collision occurred during frame reception. CL is set only if a late collision occurs or if PSMR[RSH] is enabled. Late collisions are better defined in PSMR[LCW].

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).” Data length includes the total number of frame octets (including four bytes for CRC).

Figure 24-7 shows an example of how RxBDs are used in receiving.

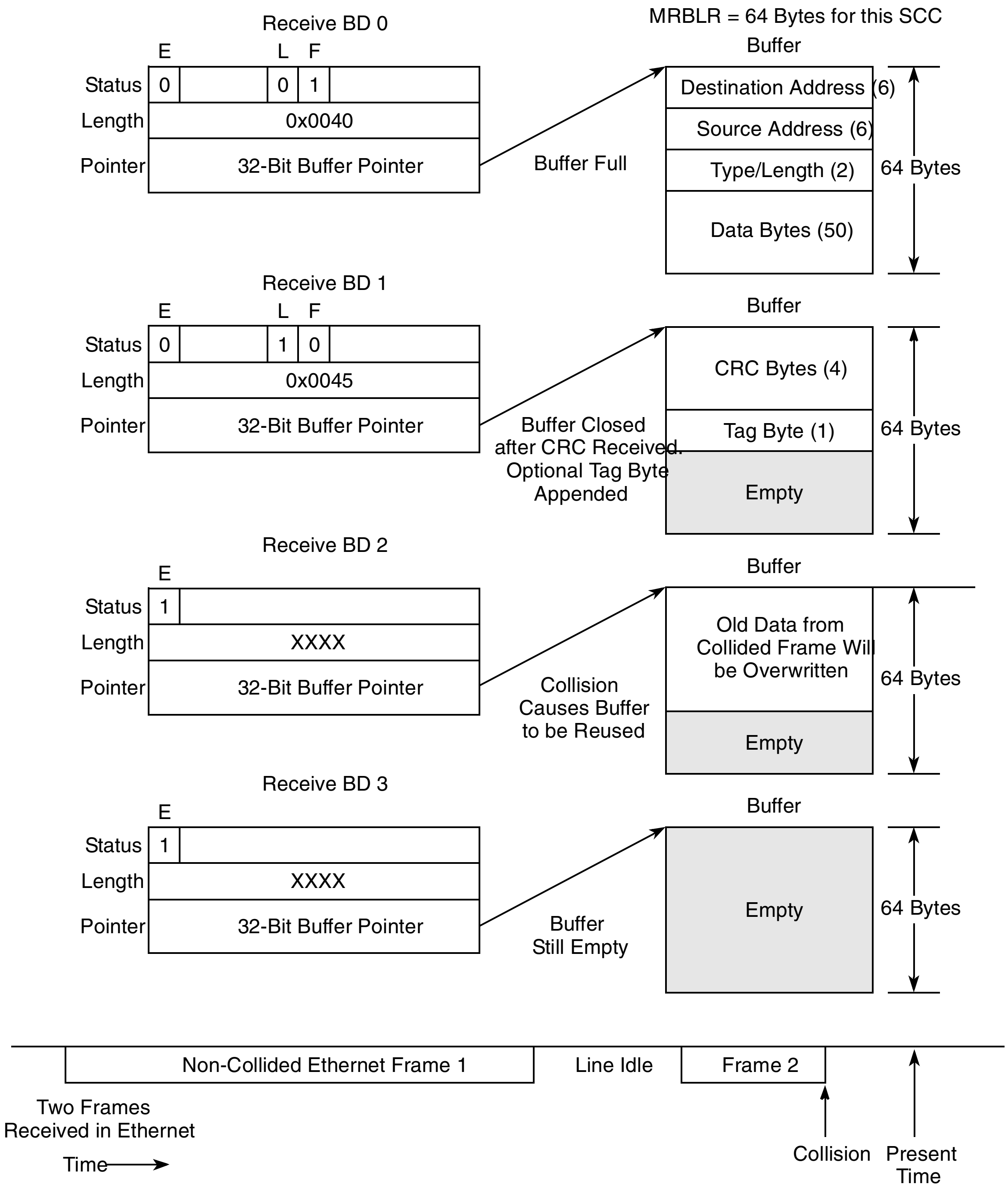


Figure 24-7. Ethernet Receiving using RxBDs

24.19 SCC Ethernet Transmit Buffer Descriptor

Data is sent to the Ethernet controller for transmission on an SCC channel by arranging it in buffers referenced by the channel TxBD table. The Ethernet controller uses TxBDs to confirm transmission or indicate errors so the core knows buffers have been serviced. Figure 24-8 represents an SCC ethernet transmit buffer descriptor.

	0	1	2	3	4	5	6	7	8	9	10	13	14	15
Offset + 0	**R**	**PAD**	**W**	**I**	**L**	**TC**	DEF	HB	LC	RL	RC		UN	CSL
Offset + 2	**Data Length**													
Offset + 4	**Tx Data Buffer Pointer**													
Offset + 6														

Figure 24-8. SCC Ethernet TxBD

Table 24-8 describes TxBD status and control fields.

Table 24-8. SCC Ethernet TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. The user can update this BD or its data buffer. The CPM clears R after the buffer has been sent or after an error occurs. 1 The user-prepared buffer has not been sent or is currently being sent. Do not modify this BD.
1	**PAD**	Short frame padding. Valid only when L is set. Otherwise, it is ignored. 0 Do not add PADs to short frames. 1 Add PADs to short frames. Pad bytes are inserted until the length of the sent frame equals the MINFLR and they are stored in PADs in the parameter RAM.
2	**W**	Wrap (final BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CPM receives incoming data into the first BD that TBASE points to in the table. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM. **Note:** The TxBD table must contain more than one BD in Ethernet mode.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is serviced; SCCE[TXE] is unaffected. 1 SCCE[TXB] or SCCE[TXE] is set after this buffer is serviced. These bits can cause interrupts if they are enabled.
4	**L**	Last. 0 Not the last buffer in the transmit frame 1 Last buffer in the transmit frame
5	**TC**	Tx CRC. Valid only when L = 1. Otherwise, it is ignored. 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. The frame was deferred before being sent successfully, that is, the transmitter had to wait for carrier sense before sending because the line was busy. This is not a collision indication; collisions are indicated in RC.
7	HB	Heartbeat. Set when the collision input was not asserted within 20 transmit clocks after transmission. HB cannot be set unless PSMR[HBC] = 1. The SCC writes HB after it finishes sending the buffer.
8	LC	Late collision. Set when a collision occurred after the number of bytes defined for PSMR[LCW] are sent. The Ethernet controller stops sending and writes this bit after it finishes sending the buffer.
9	RL	Retransmission limit. Set when the transmitter fails (Retry Limit + 1) attempts to successfully transmit a message because of repeated collisions on the medium. The Ethernet controller writes this bit after it finishes attempting to send the buffer.

Table 24-8. SCC Ethernet TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
10–13	RC	Retry count. Indicates the number of retries required before the frame was sent successfully. If RC = 0, the frame was sent correctly the first time. If RC = 15 and RET_LIM = 15 in the parameter RAM, 15 retries were required. Because the counter saturates at 15, if RC = 15 and RET_LIM > 15, 15 or more retries were required. The controller writes this field after it successfully sends the buffer.
14	UN	Underrun. Set when the Ethernet controller encounters a transmitter underrun while sending the buffer. The Ethernet controller writes UN after it finishes sending the buffer.
15	CSL	Carrier sense lost. Set when carrier sense is lost during frame transmission. The Ethernet controller writes CSL after it finishes sending the buffer.

Data length and buffer pointer fields are described in Section 19.2, “SCC Buffer Descriptors (BDs).”

24.20 SCC Ethernet Event Register (SCCE)/Mask Register (SCCM)

The SCC event register (SCCE) is used as the Ethernet event register to generate interrupts and report events recognized by the Ethernet channel. When an event is recognized, the Ethernet controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the Ethernet mask register (SCCM). SCCE bits are cleared by writing ones; writing zeros has no effect. All unmasked bits must be cleared before the CPM clears the internal interrupt request. The SCCE/SCCM registers are displayed in Figure 24-9.

	0 7	8	9 10	11	12	13	14	15
Field	—	GRA	—	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x11A10 (SCCE1); 0x11A50 (SCCE3); 0x11A70 (SCCE4) 0x11A14 (SCCM1); 0x11A54 (SCCM3); 0x11A74 (SCCM4)							

Figure 24-9. SCC Ethernet Event Register (SCCE)/Mask Register (SCCM)

Table 24-9 describes SCCE and SCCM fields.

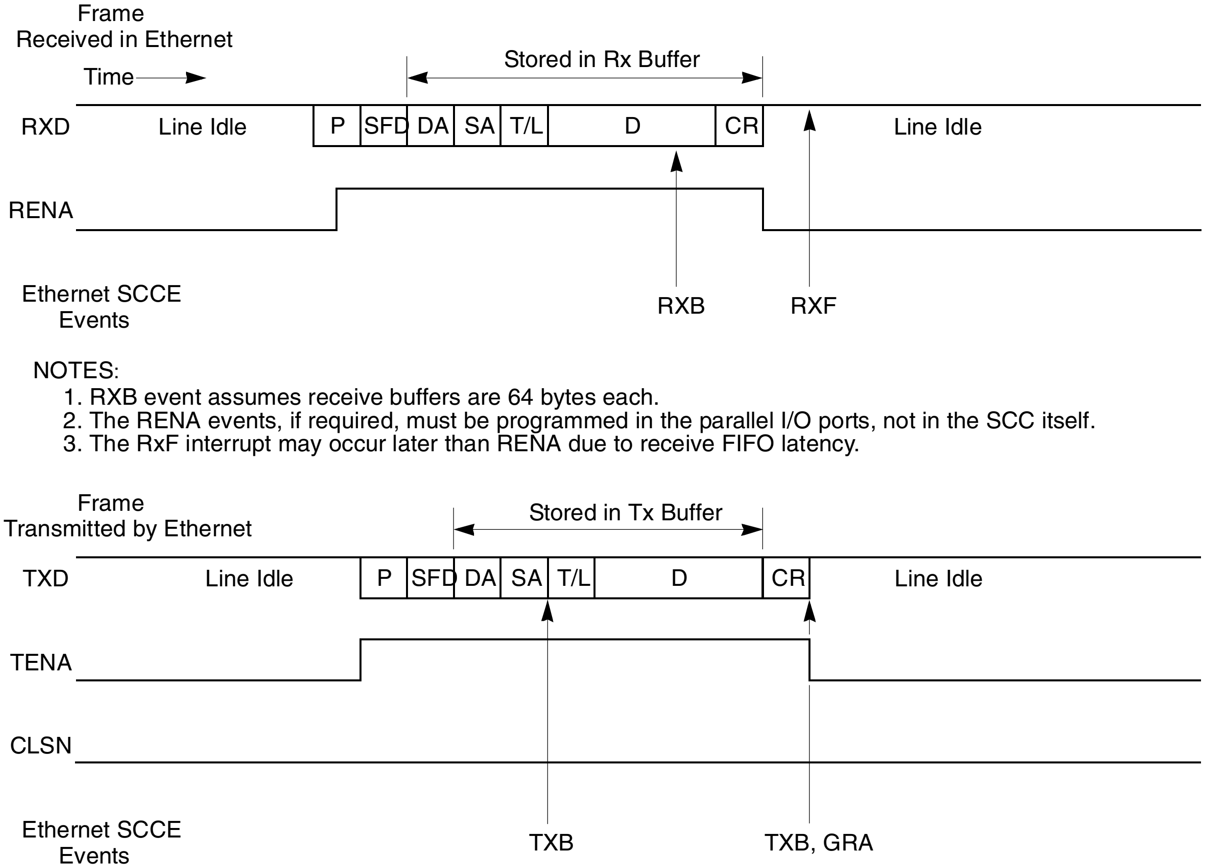
Table 24-9. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. Set as soon the transmitter finishes any frame that was in progress when a GRACEFUL STOP TRANSMIT command was issued. It is set immediately if no frame was in progress.
9–10	—	Reserved, should be cleared.
11	TXE	Set when an error occurs on the transmitter channel. This event is not maskable via the TxBD[I] bit.
12	RXF	Rx frame. Set when a complete frame has been received on the Ethernet channel.
13	BSY	Busy condition. Set when a frame is received and discarded due to a lack of buffers.

Table 24-9. SCCE/SCCM Field Descriptions (continued)

Bits	Name	Description
14	TXB	Tx buffer. Set when a buffer has been sent on the Ethernet channel.
15	RXB	Rx buffer. Set when a buffer that was not a complete frame was received on the Ethernet channel.

Figure 24-10 shows an example of interrupts that can be generated in Ethernet protocol.



NOTES:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RENA events, if required, must be programmed in the parallel I/O ports, not in the SCC itself.
3. The RxF interrupt may occur later than RENA due to receive FIFO latency.

NOTES:
1. TXB events assume the frame required two transmit buffers.
2. The GRA event assumes a GRACEFUL STOP TRANSMIT command was issued during frame transmission.
3. The TENA or CLSN events, if required, must be programmed in the parallel I/O ports, not in the SCC itself.

LEGEND:
P = Preamble, SFD = Start frame delimiter, DA and SA = Source/Destination address,
T/L = Type/Length, D = Data, CR = CRC bytes

Figure 24-10. Ethernet Interrupt Events Example

Note that the SCC status register (SCCS) cannot be used with the Ethernet protocol. The current state of the RENA and CLSN signals can be found in the parallel I/O ports.

24.21 SCC Ethernet Programming Example

The following is an initialization sequence for the SCC4 in Ethernet mode. The CLK5 pin is used for the Ethernet receiver and CLK6 is used for the transmitter.

1. Configure port D pins to enable TXD4 and RXD4. Set PPARD[21,22] and PDIRD[21] and clear PDIRD[22] and PSORD[21,22].
2. Configure ports C and D pins to enable TENA4 ($\overline{\text{RTS4}}$), CLSN4 ($\overline{\text{CTS4}}$) and RENA4 ($\overline{\text{CD4}}$). Set PPARC[8,9], PPARD[20] and PDIRD[20] and clear PDIRC[8,9], PSORC[8,9] and PSORD[20].
3. Configure port C pins to enable CLK5 and CLK6. Set PPARC[26,27] and clear PDIRC[26,27] and PSORC[26,27].
4. Connect CLK5 to the SCC4 receiver and CLK6 to the transmitter using the CPM mux. Program CMXSCR[R4CS] to 0b100 and CMXSCR[T4CS] to 0b101.
5. Connect the SCC4 to the NMSI and clear CMXSCR[SC4].
6. Write RBASE and TBASE in the SCC4 parameter RAM to point to the RxBD and TxBD in the dual-port RAM. Assuming one RxBD at the beginning of the dual-port RAM and one TxBD following that RxBD, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write 0x0CE1_0000 to the CPCR to execute an INIT RX AND TX PARAMETERS command for this channel.
8. Clear CRCEC, ALEC, and DISFC for clarity.
9. Write PAD with 0x8888 for the PAD value.
10. Write RET_LIM with 0x000F.
11. Write MFLR with 0x05EE to make the maximum frame size 1518 bytes.
12. Write MINFLR with 0x0040 to make the minimum frame size 64 bytes.
13. Write MAXD1 and MAXD2 with 0x05F0 to make the maximum DMA count 1520 bytes.
14. Clear GADDR1–GADDR4. The group hash table is not used.
15. Write PADDR1_H with 0x0000, PADDR1_M with 0x0000, and PADDR1_L with 0x0040 to configure the physical address.
16. Clear P_PER. It is not used.
17. Clear IADDR1–IADDR4. The individual hash table is not used.
18. Clear TADDR_H, TADDR_M, and TADDR_L for clarity.
19. Initialize the RxBD and assume the Rx data 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].
20. Initialize the TxBD and assume the Tx data frame is at 0x0000_2000 in main memory and contains fourteen 8-bit characters (destination and source addresses plus the type field). Write 0xFC00 to TxBD[Status and Control], add PAD to the frame and generate a CRC. Then write 0x000D to TxBD[Data Length] and 0x0000_2000 to TxBD[Buffer Pointer].
21. Write 0xFFFF to the SCCE register to clear any previous events.
22. Write 0x001A to the SCCM register to enable the TXE, RXF, and TXB interrupts.

23. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
24. Write 0x0000_0000 to GSMR_H4 to enable normal operation of all modes.
25. Write 0x1088_000C to the GSMR_L4 register to configure $\overline{\text{CTS}}$ (CLSN) and $\overline{\text{CD}}$ (RENA) to automatically control transmission and reception (DIAG bits) and the Ethernet mode. TCI is set to allow more setup time for the EEST to receive the MPC8272 transmit data. TPL and TPP are set for Ethernet requirements. The DPLL is not used with Ethernet. Note that the ENT and ENR are not enabled yet.
26. Write 0xD555 to the DSR.
27. Set the PSMR4 to 0x0A0A to configure 32-bit CRC, promiscuous mode, and begin searching for the start frame delimiter 22 bits after RENA4 ($\overline{\text{CD4}}$).
28. Write 0x1088_003C to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last.

After 14 bytes and the 46 bytes of automatic pad (plus the 4 bytes of CRC) are sent, the TxBD is closed. Additionally, the receive buffer is closed after a frame is received. Any data received after 1520 bytes or a single frame causes a busy (out-of-buffers) condition because only one RxBD is prepared.

Chapter 25
SCC AppleTalk Mode

AppleTalk is a set of protocols developed by Apple Computer, Inc. to provide a LAN service between Macintosh computers and printers. Although AppleTalk can be implemented over a variety of physical and link layers, including Ethernet, AppleTalk protocols have been most closely associated with the LocalTalk physical and link-layer protocol, an HDLC-based protocol that runs at 230.4 Kbps. In this manual, the term ‘AppleTalk controller’ refers to the support that the MPC8272 provides for the LocalTalk protocol. The AppleTalk controller provides requires frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames. These capabilities, with the use of the HDLC controller in conjunction with DPLL operation in FM0 mode, provide the proper connection formats to the LocalTalk bus.

25.1 Operating the LocalTalk Bus

A LocalTalk frame, shown in Figure 25-1, is basically a modified HDLC frame.

Sync Sequence	HDLC Flags	Destination Address	Source Address	Control Byte	Data (Optional)	CRC-16	Closing Flag	Abort Sequence
> 3 bits	2 or more bytes	1 byte	1 byte	1 byte	0–600 bytes	2 bytes	1 byte	12–18 ones

Figure 25-1. LocalTalk Frame Format

First, a synchronization sequence of more than 3 bits is sent. This sequence consists of at least 1 logical one bit (FM0 encoded) followed by 2 bit times or more of line idle with no particular maximum time specified. The idle time allows LocalTalk equipment to sense a carrier by detecting a missing clock on the line. The remainder of the frame is a typical half-duplex HDLC frame. Two or more flags are sent, allowing bit, byte, and frame delineation or detection. Two bytes of address, destination, and source are sent next, followed by a byte of control and 0–600 data bytes. Next, two bytes of CRC (the common 16-bit CRC-CCITT polynomial referenced in the HDLC standard protocol) are sent. The LocalTalk frame is then terminated by a flag and a restricted HDLC abort sequence. Finally, the transmitter’s driver is disabled.

The control byte within the LocalTalk frame indicates the type of frame. Control byte values from 0x01–0x7F are data frames; control byte values from 0x80–0xFF are control frames. Four control frames are defined:

- ENQ—Enquiry
- ACK—Enquiry acknowledgment
- RTS—Request to send a data frame
- CTS—Clear to send a data frame

Frames are sent in groups known as dialogs, which are handled by the software. For instance, to transfer a data frame, three frames are sent over the network. An RTS frame (not to be confused with the RS-232

$\overline{\text{RTS}}$ pin) is sent to request the network, a CTS frame is sent by the destination node, and the data frame is sent by the requesting node. These three frames comprise one possible type of dialog. After a dialog begins, other nodes cannot start sending until the dialog is complete. Frames within a dialog are sent with a maximum interframe gap (IFG) of 200 µs. Although the LocalTalk specification does not state it, there is also a minimum recommended IFG of 50 µs. Dialogs must be separated by a minimum interdialog gap (IDG) of 400 µs. In general, these gaps are implemented by the software.

Depending on the protocol, collisions should be encountered only during RTS and ENQ frames. Once frame transmission begins, it is fully sent, regardless of whether it collides with another frame. ENQ frames are infrequent and are sent only when a node powers up and enters the network. A higher-level protocol controls the uniqueness and transmission of ENQ frames.

In addition to the frame fields, LocalTalk requires that the frame be FM0 (differential Manchester space) encoded, which requires one level transition on every bit boundary. If the value to be encoded is a logical zero, FM0 requires a second transition in the middle of the bit time. The purpose of FM0 encoding is to avoid having to transmit clocking information on a separate wire. With FM0, the clocking information is present whenever valid data is present.

25.2 Features

The following list summarizes the features of the SCC in AppleTalk mode:

- Superset of the HDLC controller features
- FM0 encoding/decoding
- Programmable transmission of sync sequence
- Automatic postamble transmission
- Reception of sync sequence does not cause extra SCCE[DCC] interrupts
- Reception is automatically disabled while sending a frame
- Transmit-on-demand feature expedites frames
- Connects directly to an RS-422 transceiver

25.3 Connecting to AppleTalk

As shown in Figure 25-2, the MPC8272 connects to LocalTalk, and, using TXD, $\overline{\text{RTS}}$, and RXD, serves as an interface for the RS-422 transceiver. The RS-422, in turn, is an interface for the LocalTalk connector. Although it is not shown, a passive RC circuit is recommended between the transceiver and connector.

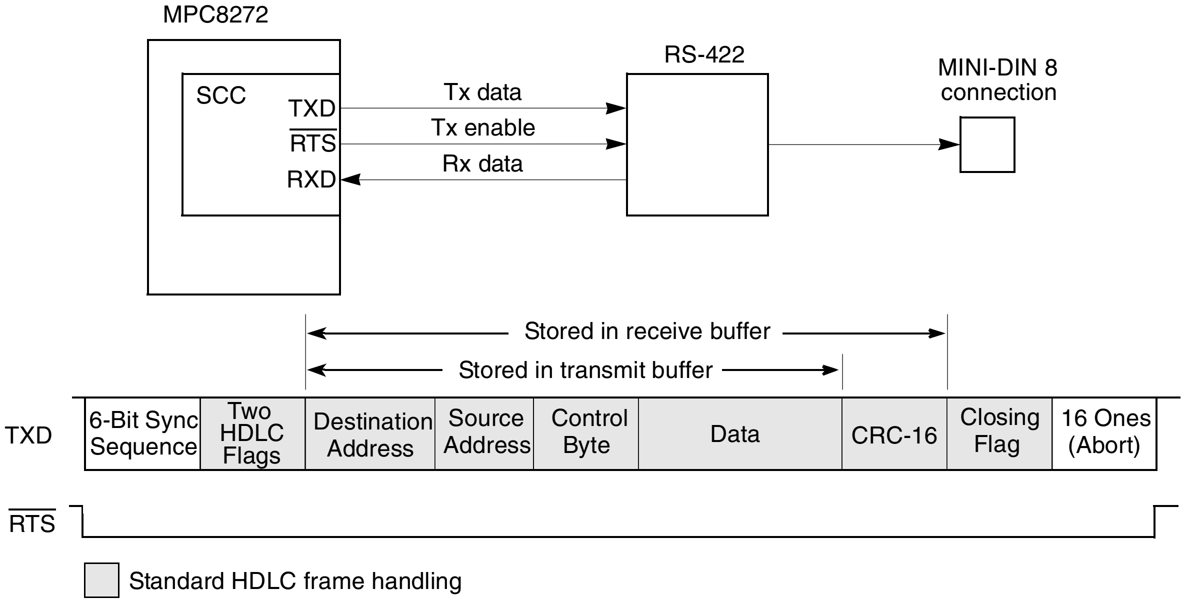


Figure 25-2. Connecting the MPC8272 to LocalTalk

The 16× overspeed of a 3.686-MHz clock can be generated from an external frequency source or from one of the baud rate generators if the resulting output frequency is close to a multiple of the 3.686-MHz frequency. The MPC8272 asserts $\overline{\text{RTS}}$ throughout the duration of the frame so that $\overline{\text{RTS}}$ can be used to enable the RS-422 transmit driver.

25.4 Programming the SCC in AppleTalk Mode

The AppleTalk controller is implemented by setting certain bits in the HDLC controller. Otherwise, Chapter 21, “SCC HDLC Mode,” describes how to program the HDLC controller. Use GSMR, PSMR, or TODR to program the AppleTalk controller.

25.4.1 Programming the GSMR

Program the GSMR as described below:

1. Set MODE to 0b0010 (AppleTalk).
2. Set DIAG to 0b00 for normal operation, with $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ grounded or configured for parallel I/O. This causes $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ to be internally asserted to the SCC.
3. Set RDCR and TDCR to (0b10) a 16× clock.
4. Set the TENC and RENC bits to 0b010 (FM0).
5. Clear TEND for default operation.
6. Set TPP to 0b11 for a preamble pattern of all ones.
7. Set TPL to 0b000 to transmit the next frame with no synchronization sequence and to 001 to transmit the next frame with the LocalTalk synchronization sequence. For example, data frames do not require a preceding synchronization sequence. These bits may be modified on-the-fly if the AppleTalk protocol is selected.

8. Clear TINV and RINV so data will not be inverted.
9. Set TSNC to 1.5 bit times (0b10).
10. Clear EDGE. Both the positive and negative edges are used to change the sample point (default).
11. Clear RTSM (default).
12. Set all other bits to zero or default.
13. Set ENT and ENR as the last step to begin operation.

25.4.2 Programming the PSMR

Follow these steps to program the protocol-specific mode register:

1. Set NOF to 0b0001 giving two flags before frames (one opening flag, plus one additional flag).
2. Set CRC 16-bit CRC-CCITT.
3. Set DRT.
4. Set all other bits to zero or default.

For the PSMR definition, see Section 21.8, “HDLC Mode Register (PSMR).”

25.4.3 Programming the TODR

Use the transmit-on-demand (TODR) register to expedite a transmit frame. See Section 19.1.4, “Transmit-on-Demand Register (TODR).”

25.4.4 SCC AppleTalk Programming Example

Except for the previously discussed register programming, use the example in Section 21.15.6, “HDLC Bus Protocol Programming.”

Chapter 26
QMC (QUICC Multi-Channel Controller)

The QUICC multi-channel controller (QMC) functionality can emulate up to 64 time-division serial channels, using a serial communication controller (SCC), and a time-division-multiplexed (TDM) physical interface. Each of the QMC channels can be independently programmed to perform in either HDLC or transparent mode.

Any available TDM in the serial interface (SI) can be used for the QMC protocol. The SI transfers data between the TDM interface and the SCC. The SCC follows by performing multiplexing/demultiplexing on the QMC channels. Figure 26-1 provides an overview of the QMC functionality.

Each SCC can work in QMC mode, either alone or together in any combination, spreading any of the 64 available QMC channels across the multiple SCCs. One TDM connection can be routed to one or more SCCs operating in QMC mode, with each SCC operating on different time slots.

It is also possible to use multiple TDMs for QMC with combined routing to one SCC or to separate SCCs. When using multiple TDMs connected to the same SCC, restrictions such as using common clocks and sync inputs apply; it is also important to avoid collisions by separating the serial interface (SI) routing, making sure that only one TDM will be accessing the SCC at any given time.

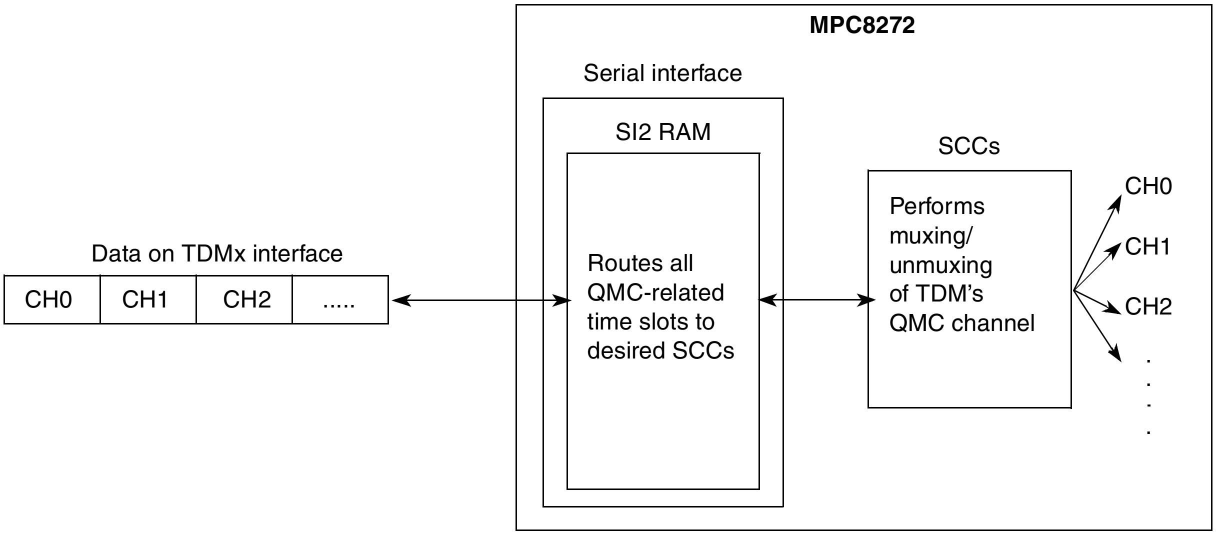

MPC8272
Serial interface
SI2 RAM
Routes all QMC-related time slots to desired SCCs
SCCs
Performs muxing/ unmuxing of TDM’s QMC channel
CH0
CH1
CH2
Data on TDMx interface
CH0
CH1
CH2
.....

Figure 26-1. QMC Channel Addressing Capability

26.1 Features

QMC-specific features include the following:

- Up to 64 independent communication channels on one SCC, or split across multiple SCCs (64 channels total per device)

- Arbitrary mapping of any of 0–63 channels to any TDM time slot
- Supports up to two simultaneous 32-channel E1 links
- Independent mapping possible for receive/transmit
- Supports either transparent or HDLC protocols for each channel
- Interrupt circular buffer with programmable size and overflow identification
- Global loop mode
- Individual channel loop mode through the SI
- Programmable frame length through the SI

QMC features related to the serial interface include the following:

- Serial-multiplexed (full duplex) input/output 2048-, 1544-, or 1536-Kbps PCM highways
- Compatible with T1/DS1 24-channel and CEPT E1 32-channel PCM highway, ISDN basic rate, ISDN primary rate and user-defined
- Subchanneling on each time slot
- Allows independent transmit and receive routing, frame syncs, and clocking
- Concatenation of any, not necessarily consecutive, time slots to channels independently for receive/transmit
- Supports H0, H11, and H12 ISDN channels
- Allows dynamic allocation of channels

QMC features related to the system interface include the following:

- On-chip bus arbitration for serial DMAs with no performance penalty
- Efficient bus usage (no bus usage for nonactive channels and active channels that have nothing to transmit)
- Efficient control of the interrupts to the CPU
- Supports external buffer descriptors table
- Uses on-chip enlarged dual-ported RAM for parameter storage

26.2 QMC and the Serial Interface

QMC is designed to work in conjunction with the serial interface, taking advantage of its programmable SIRAM and additional functionalities. See Chapter 14, “Serial Interface with Time-Slot Assigner,” for details on proper programming of the SI’s registers and SIRAM. However, it is possible to operate QMC in non-multiplexed serial interface (NMSI) mode, directly using the SCC’s own pins instead of the TDM interlace pins. Functions such as frame synchronization, loopback, echo, and inverted signals are performed in the serial interface and cannot be achieved in NMSI mode. It is recommended to use the serial interface even if only one SCC is used for the TDM bus.

Connecting an SCC to the SI or to its own pins in NMSI mode is selected by programming the CMX SCC clock route register (CMXSCR). See Section 15.4.4, “CMX SCC Clock Route Register (CMXSCR),” for details on proper programming of the CMXSCR.

This section describes additional functionality that the SI provides to QMC operation.

26.2.1 Synchronization

Independent receive and transmit clocks and frame synchronization signals control the data transfer. In NMSI operation, synchronization occurs only once after activating QMC, to initiate transfer using the CD (receive) and CTS (transmit) signals in pulse mode. If any noise corrupts either signal or the clock, the QMC will be out of synchronization until the whole protocol is restarted.

In contrast, the more robust SI performs a synchronization on each frame, limiting the damage from noise error on the clock or synchronization lines. Noisy channels can be restarted individually without interrupting other channels.

26.2.2 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.

26.2.3 Echo Mode

The SI can be programmed to echo incoming data. In this mode, the complete TDM link is retransmitted from the incoming L1RXDx to the L1TXDx pin on a bit-by-bit basis. The receiver section of the selected SCC can operate normally and also receive the incoming bit stream. This is also known as global echo mode on the whole link. Individual time-slot echo is not possible with QMC without software intervention.

26.2.4 Inverted Signals

All QMC-related receive and transmit data can be logically inverted by setting the RINV and TINV bits of the GSMR_L register. A logical inversion on a per channel basis is not possible in the QMC without external hardware. To invert a specific channel, the SI can be programmed to send a strobe signal at the QMC channel’s corresponding time slot on the TDM interface. This strobe can then be connected to an external XOR gate to perform the inversion.

26.2.5 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.

26.3 QMC Memory Organization

26.3.1 QMC Memory Structure

Figure 26-2 shows how data is addressed by the QMC protocol. It discusses addressing the dual-ported RAM to access data within the buffers.

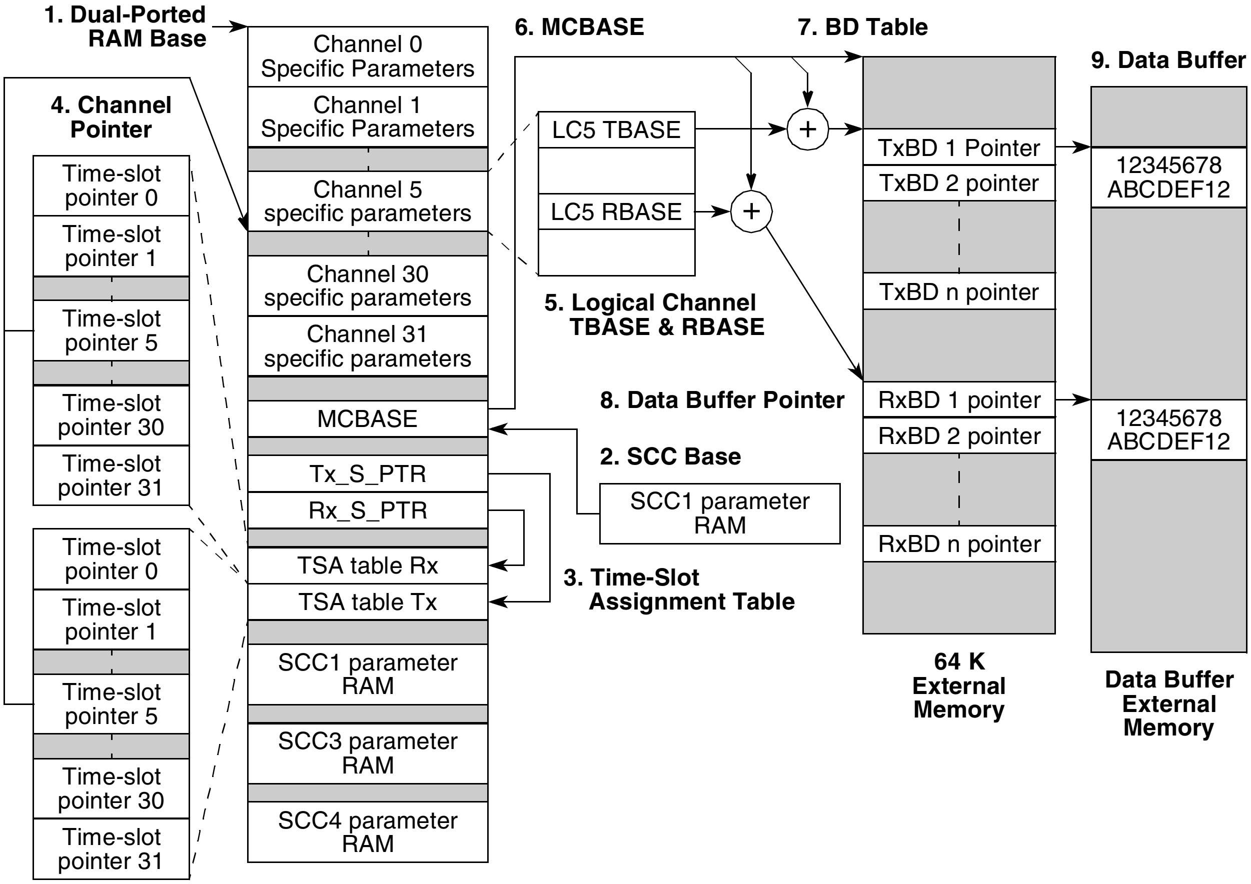


Figure 26-2. QMC Memory Structure

26.3.2 SCC Base and Global Multichannel Parameters

The SCC base points to the start of the parameter RAM for each of the SCCs at 256-byte intervals. When the QMC protocol is enabled on an SCC, its parameter RAM is used to store the global multichannel parameters for all the logical channels. This area contains parameters and pointers that are common to all channels.

26.3.2.1 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 SCC Base + 20; this is the normal location of the receive time slot assignment table. The Tx_S_PTR is normally set to SCC Base + 60; this is the normal location of the transmit time-slot assignment table. However, if the receiver and the transmitter have the same mapping for the logical channels, Tx_S_PTR can point to SCC base + 20 so that Rx and Tx have a common time-slot assignment table. Note that if a single TDM channel is routed to more than one SCC, they may also use just one time-slot assignment table for all SCCs. See Section 26.3.3, “Multiple SCC Assignment Tables,” for more information. The time-slot assignment table holds one 32-bit entry for each time slot. It has options for subchanneling, a valid bit, and a logical channel pointer. For 64-channel support there is only space for one table; therefore, common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC. Associated with the Rx/Tx_S_PTR are the Rx/TxPTR pointers that are maintained by the CPM and point to the current time slot.

26.3.2.2 TSATRx/TSATTx Channel Pointers

The channel pointers are 12-bit pointers to the channel-specific parameters in the internal dual-ported RAM. These should not be confused with TSATRx/TSATTx pointers as described in Section 26.3.2.1, "TSATRx/TSATTx Pointers and Time-Slot Assignment Table." The 6 most-significant bits of the address are taken from the time slot assignment table. The 6 least-significant bits are zero, mapping out a 64-byte area for each of the channel-specific parameters. The channel-specific parameters are common for Rx and Tx. For 32-channel support, 2 Kbytes of dual-ported RAM is required (32 × 64), and for 64-channel support, 4 Kbytes of dual-ported RAM is required (64 × 64). In most cases, time slot 0 channel pointer will address the base of dual-ported RAM for logical channel 0, and time slot 1 channel pointer would address the base of dual-ported RAM + 4 for logical channel 1. In Figure 26-2, time slot 5 channel pointer addresses logical channel 5, requiring the channel pointer being set to 0b000101.

NOTE

It is possible to concatenate multiple time slots to one logical channel. This is achieved by setting the channel pointers of the grouped time slots to the same logical channel.

26.3.2.3 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.

26.3.2.4 MCBASE

MCBASE is located in the global multichannel parameters. Each SCC has a unique MCBASE value pointing to the base of the SCC's buffer descriptor table in external memory. For example, the address of logical channel five's Tx buffer descriptor table is MCBASE + logical channel five TBASE. MCBASE normally points to external RAM, but it is permissible to set it up so that some or all BDs are placed within free areas of the DPRAM. If any BDs are placed in internal memory, the GBL bit of RSTATE and TSTATE may not be set. This may save valuable access time if external memory is slow.

26.3.2.5 Buffer Descriptor Table

A buffer descriptor table for each SCC is located in a 64-Kbyte area of external memory. This block size is determined by the TBASE and RBASE addressing range. The memory segment must be long-word-aligned but can start anywhere in memory. Each SCC has a maximum of 16,384 (64 Kbytes memory ÷ 4-byte pointers) buffers. For a 32-channel implementation, each logical channel has a maximum of 256 (16,384 / (32 × 2)) buffers for receive and 256 buffers for transmit. For each logical channel, a circular queue exists with programmable start address and length.

26.3.2.6 Data Buffer Pointer

As with the standard CPM protocols, the data buffer is addressed by a 32-bit pointer within the buffer descriptor. This addresses the data received or transmitted from external memory.

26.3.2.7 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.

26.3.2.8 Global Multichannel Parameters

The global multichannel parameters reside in the SCC's parameter RAM page and are common to all logical channels.

The largest portion of the global area is the time-slot assigner tables for the receiver and transmitter section of the SCC. For 32-channel support, there is one table for Tx and one for Rx within the parameter RAM. If the connection is split over multiple SCCs, this table only needs to be present once for multiple SCCs operating in QMC mode. See Section 26.3.3, "Multiple SCC Assignment Tables," for more information. For 64-channel support there is only space for one table; therefore common Rx and Tx parameters will need to be used unless one of the TSA tables can be accommodated elsewhere in memory, such as in the parameter RAM area of another SCC.

The dual-ported RAM is used for the channel-specific area for all SCCs. It is important that individual time slots are mapped to only one SCC, and that individual logical channels are separated to avoid contention.

Table 26-1 lists the global parameters. Note that the boldfaced parameters must be initialized by the user. See Section 26.7, "QMC Initialization," for more information.

Table 26-1. Global Multichannel Parameters

Offset to SCC 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.
06	**MRBLR**	16	Maximum receive buffer length—This host-initialized entry defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. This parameter is only valid in HDLC mode. The buffer area allocated in memory for each buffer is MRBLR + 4. The QMC adds another long word if non-octet-aligned frames are received in HDLC operation. The non-octet information is written only to the last buffer of a frame, but it can happen in any buffer. See Section 26.6.1, "Receive Buffer Descriptor," for more information. As the QMC works on long-word alignment, MRBLR value should be a multiple of 4 bytes.

Table 26-1. Global Multichannel Parameters (continued)

Offset to SCC Base	Name	Width (Bits)	Description
08	**Tx_S_PTR**	16	Tx time-slot assignment table pointer (SCC base + 60 in normal mode; SCC base + 20 for common Rx & Tx time slot assignment tables)—This global QMC parameter defines the start value of the TSATTx table. The TSATTx table in the global multichannel parameter listing starts by default at SCC base + 60. Tx_S_PTR lets the user move the starting address of this table. If the same routing and masking are used for the transmitter and receiver, the tables can be overlaid, so Tx_S_PTR can point to SCC base + 20. This parameter is an offset from DPRBASE. This table must be present only once per SCC global area. Other SCCs can access this location.
0A	**RxPTR**	16	Rx pointer (initialize to SCC base + 20)—This global QMC parameter is a RISC variable that points to the current receiver time slot. The host must initialize this pointer to the starting location of TSATRx. The RISC processor increments this pointer whenever it completes the processing of a received time slot.
0C	**GRFTHR**	16	Global receive frame threshold—Used to reduce interrupt overhead when many short HDLC frames arrive, each causing an RXF interrupt. GRFTHR can be set to limit the frequency of interrupts. Set to 1 to get an interrupt per frame received. Note that the RXF event is written to the interrupt table on each received frame, but GINT is set only when the number of RXF events (by all channels) reaches the GRFTHR value. GRFTHR can be changed on the fly. For information about exception handling, see Section 26.5, “QMC Exceptions.”
0E	**GRFCNT**	16	Global receive frame count (initialized GRFCNT = GRFTHR)—A down-counter used to implement the GRFTHR feature. GRFCNT decrements for each frame received. No other receiver interrupts affect this counter. The counter value is set to the threshold during initialization. GRFCNT is automatically reset to the GRFTHR value by the CPM after a global interrupt.
10	**INTBASE**	32	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 SCC operating in QMC mode has its own interrupt table in external memory. See Section 26.5, “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 = SCC base + 20 in normal mode)—This global QMC parameter defines the start value of the TSATRx table, which must be present only once per SCC global area. Other SCCs may access this location.
1A	**TxPTR**	16	TxPTR (initialize to SCC Base + 60)—This global parameter is a RISC variable that points to the current transmitter time slot. The host must initialize it to the starting location of TSATTx. The RISC processor increments this pointer whenever it completes the processing of a transmitter time slot.

Table 26-1. Global Multichannel Parameters (continued)

Offset to SCC 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 26.3.4.1, "Channel-Specific HDLC Parameters." This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
20	**TSATRx**	32 Entries x 16	Time slot assignment table Rx—Host-initialized, 16-bit-wide table with 32 entries that define mapping of logical channels to time slots for the QMC receiver. The QMC protocol looks at chunks of 8 bits regardless of whether they come from one physical time slot of the TDM or whatever other combination of bits the TSA transfers to the SCC. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time-slot assignment table. See Figure 26-4.
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 SCC to the TDM through the TSA. These 8 bits are referred to as a time slot in the assignment table. It is recommended but not required to route all bits from the TDM to the SCC and to do all enabling and masking in the time slot assignment table. See Figure 26-4.
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 26.3.4.1, "Channel-Specific HDLC Parameters." This entry must have a correct value if at least one HDLC channel is used; otherwise, it can be cleared (0).
A4	TEMP_RBA	32	Temporary receive buffer address
A8	TEMP_CRC	32	Temporary cyclic redundancy check

Table 26-1. Global Multichannel Parameters (continued)

Offset to SCC Base	Name	Width (Bits)	Description
AC	RX_FRM_Base	16	This entry contains a pointer to an area in the DPR where the receiver framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel. When the QMC is handling up to 32 channels it is possible to program this entry to the value SCC Base + 0xC0, thus locating this data structure inside the SCC parameter page and saving DPR space. When using 64 channels this entry should point to a free area of 64 bytes in a DPR that is 64 bytes aligned.
AE	TX_FRM_Base	16	This entry contains a pointer to an area in the DPR where the receiver framer temporary parameters reside. The area designated for these parameters is a single byte per TDM channel. When the QMC is handling up to 32 channels it is possible to program this entry to the value SCC Base + 0xE0, thus locating this data structure inside the SCC parameter page and saving DPR space. When using 64 channels this entry should point to a free area of 64 bytes in a DPR that is 64 bytes aligned.

NOTE

There is no way the receiver and transmitter can share the same structure as done on the TSA table when the TDM functionality of the receiver is identical to this of the transmitter.

An area of 64 bytes starting at address SCC Base + 0xC0 up to SCC Base + 0xFF is available for this data structure.

The user must program the area pointed by RX_FRM_Base / TX_FRM_Base as follow:

Each 1-byte entry corresponds to a TDM channel numbered per the QMC time slot assignment table. The number of entries is determined by the number of active channels in the system.

NOTE

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	1

Figure 26-3. RX Framer Entry

Each time a channel is initialized, the corresponding entry should be programmed to this value.

NOTE

The area between SCC base + 20 and SCC base + 9F is normally used for TSA tables. The mapping above is ideal for 32-channel support. The exact mapping of the TSA tables is determined by the programming of Rx_S_PTR and Tx_S_PTR, and is not fixed. For 64-channel support it is suggested to use common Rx and Tx parameters. The TSA table will be common and have 64 entries starting at SCC base + 20; see Figure 26-5. Alternatively, another SCC's parameter RAM may be used, as determined by Rx_S_PTR and Tx_S_PTR; see Figure 26-7 for more information. However implemented, the TSA tables may reside anywhere in internal memory.

Figure 26-4 shows a general time-slot assignment table for 32 16-bit time slots. The fields will be used to either transmit or receive channels.

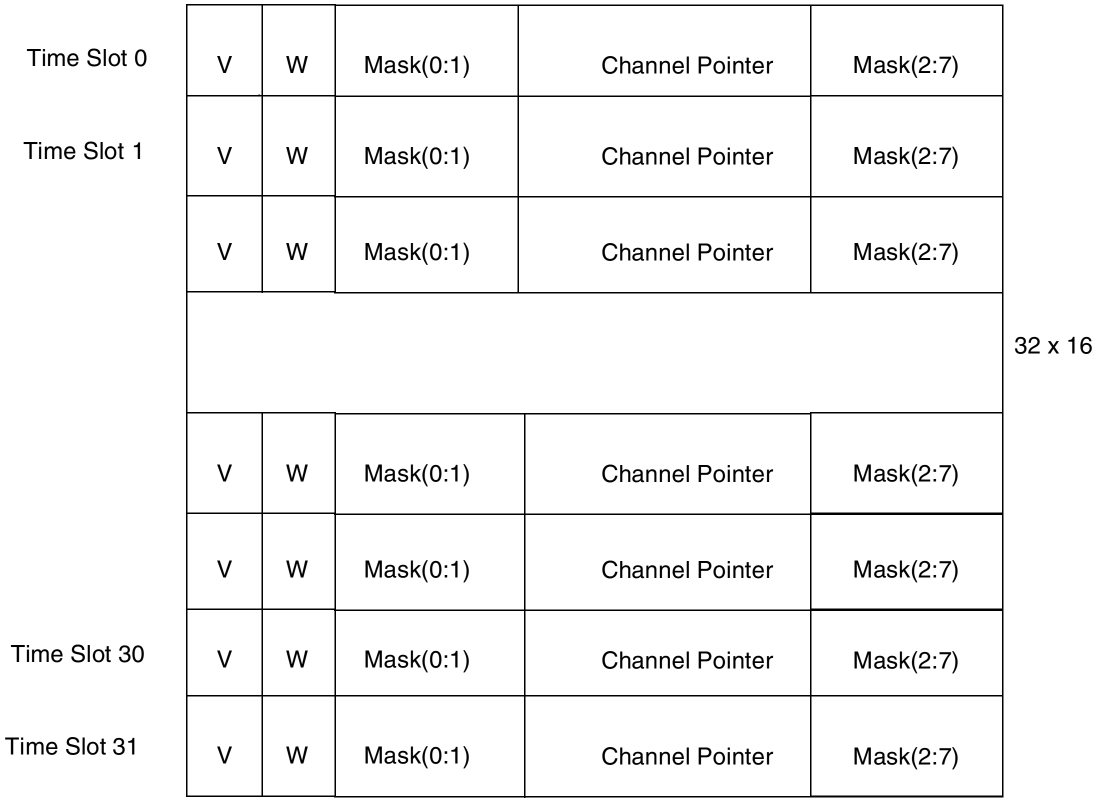


Figure 26-4. Time-Slot Assignment Table

Table 26-2 describes the fields in the time-slot assignment table for receive.

Table 26-2. Time-Slot Assignment Table Entry Fields for Receive Section

Field	Description
V	Valid bit—The valid bit indicates whether this time slot is valid. • 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 26-3 describes the fields in the time-slot assignment table for transmit.

Table 26-3. Time-Slot Assignment Table Entry Fields for Transmit Section

Name	Description
V	• Valid bit—The valid bit indicates whether this time slot is valid. • 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—The 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 SCC in the next request. The next request is identified by a frame synchronization pulse.
Tx channel pointer	This 6-bit field of the TSATTx entry identifies the data channel routed to this time slot. The actual channel pointer is 12 bits long, and contains the starting address of the channel-specific parameter area (address of TBASE). The 6 most-significant bits are taken from the TSATTx channel pointer field, and the 6 least-significant bits are always internally set to zero.
Mask(0–7)	Mask bits—Identifies the valid bits in this time slot for subchanneling support. For 8-bit resolution, all mask bits should be set to 1. For a valid channel with an unmasked bit (1), the bit position is filled according to the protocol. A valid channel with a masked bit (0) transmits a logic high (1).

If the transmitter and receiver have the same mapping, it is possible to use a common time-slot assignment table. This is initialized by setting both Tx_S_PTR and Rx_S_PTR to SCC Base + 20. For 64-channel support it is suggested to use common Rx and Tx parameters. The time slot assignment table will then also be common and have 64 entries starting at SCC Base + 20; see Figure 26-5.

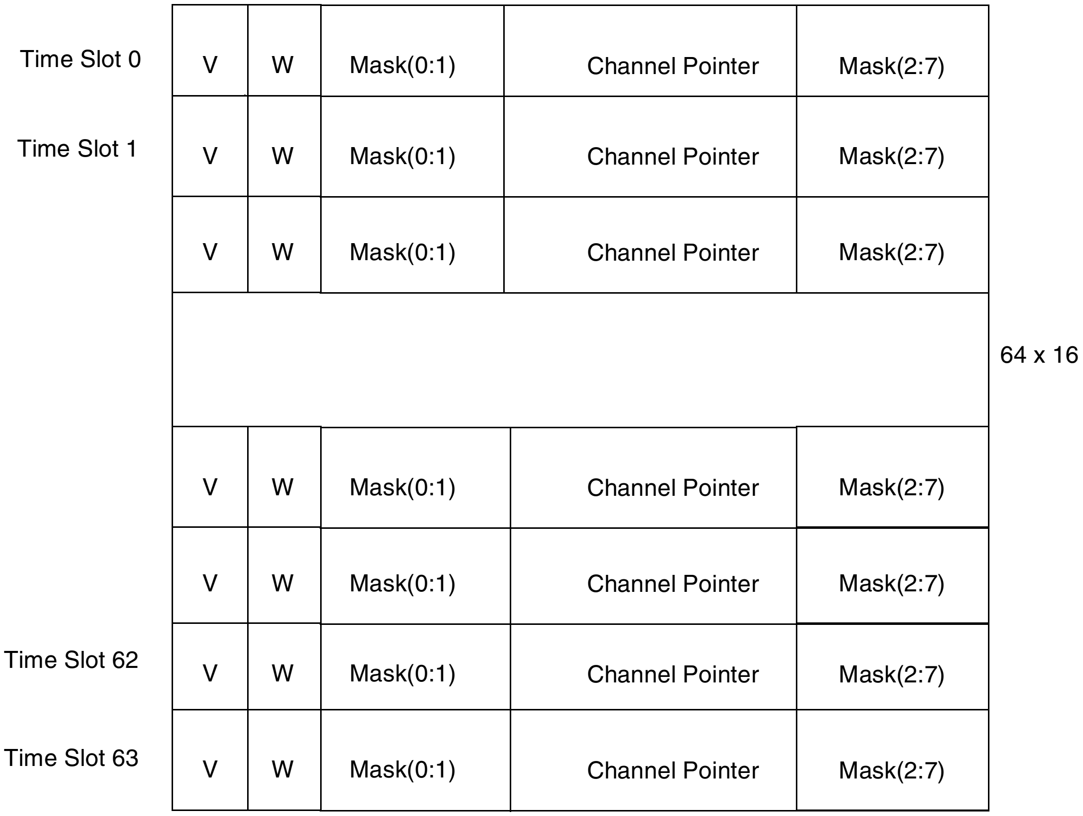


Figure 26-5. Time Slot Assignment Table for 64-Channel Common Rx and Tx Mapping

26.3.3 Multiple SCC Assignment Tables

Assume a scenario as depicted in Figure 26-6. A 2.048-Mbps TDM link is fed directly into the TSA. Within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC1. This arrangement is used to spread the load over two SCCs. This effectively doubles the FIFO depth on the QMC protocol. Time slots are switched to alternate SCCs to avoid data bursts that may stress the FIFOs. Each SCC sees a continuous bit stream without any gaps as described earlier.

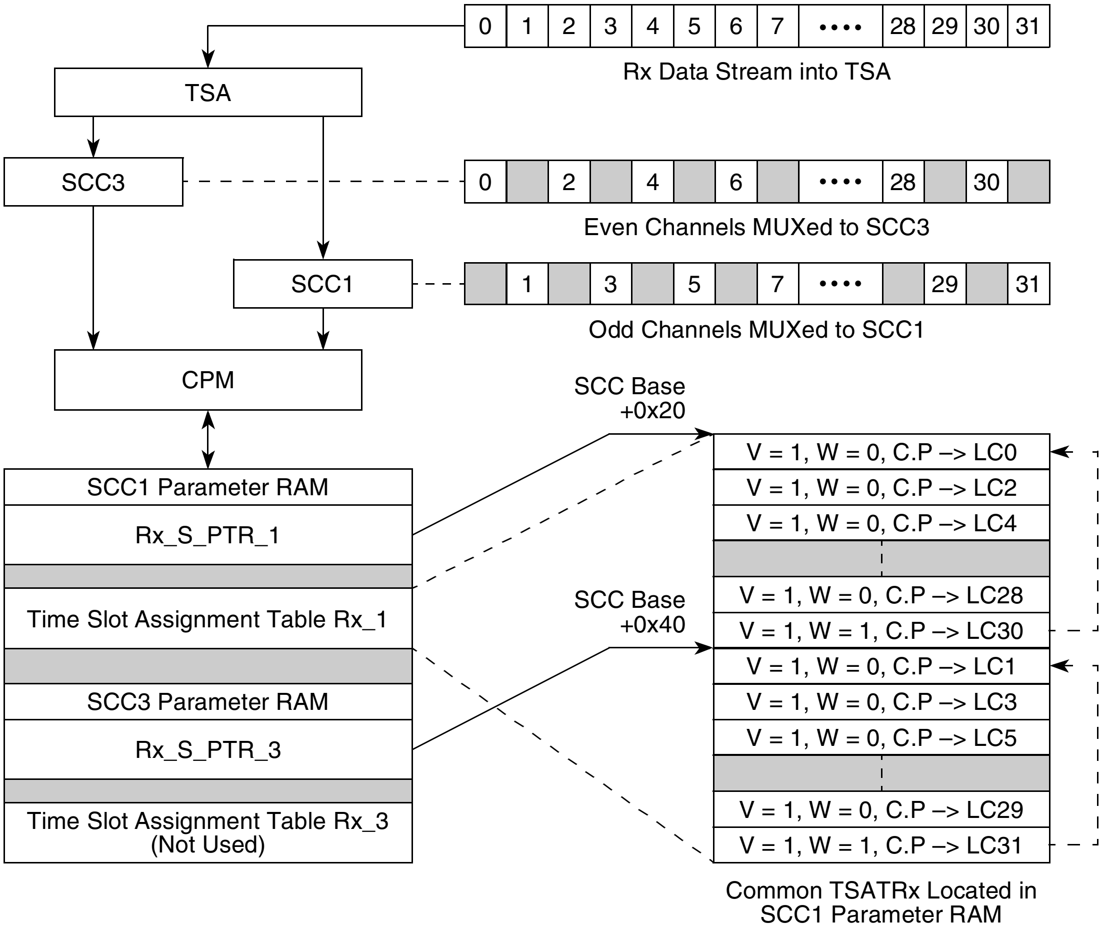


Figure 26-6. Rx Time Slot Assignment Table for 32 Channels over Two SCCs

NOTE

It is important that multiples of bytes are routed to each SCC to delineate between time slots. Unused bits shall be routed to the SCC and be masked in the time-slot assignment table.

In Figure 26-6, each SCC has its own pointer, Rx_S_PTR_1 and Rx_S_PTR_3, addressing SCC1's time-slot assignment table. This table only needs to be present once in one of the SCC1's global parameter area. Rx_S_PTR_1 points to the start of the table, address SCC base + 20. The 16 logical channels from SCC1 are located in the first 16 entries of the table. The entry for logical channel 30 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 0 on reception of the next byte routed to SCC1. Rx_S_PTR_3 addresses SCC base + 40, the start of the 16 entries for SCC3. The entry for logical channel 31 has the wrap bit (W) set, causing the CPM to wrap back to logical channel 1 on reception of the next byte routed to SCC3. Each entry within the table has a channel pointer to a logical channel. It is important that different SCCs do not point to the same logical channel.

The TSATTx is also located in SCC1's parameter RAM. This means that the area reserved for the TSA tables in SCC3's parameter RAM is free for alternative use.

A second scenario is depicted in Figure 26-7. A 4.096-Mbps TDM link is fed directly into the TSA. Again, within the SI RAM, the even channels (byte-wide) are muxed to SCC3 and the odd channels are muxed to SCC1. This arrangement is used to spread the load over two SCCs. Another reason this method may be used is to facilitate separate routing for the Rx and Tx logical channels. This requires two 64-entry tables that require 256 bytes, but only 128 bytes are allocated in the parameter RAM of an SCC for time-slot assignment tables. In this case, the Rx table is located in SCC1's parameter RAM, and the TX table is located in SCC3's parameter RAM, making most efficient use of memory.

Changes on the fly are easily accomplished by setting or clearing the valid bit for each time slot. Changes to the mask bits can also be made on the fly. This does not cause any problems to the QMC microcode itself, but may cause protocol errors on the channel in question depending on when this change is done.

It is possible to have a time-slot assignment table for every SCC in its corresponding RAM page and have all of the TDM routed to the different SCCs. This gives the user a very flexible system that can be changed on the fly without disconnecting the TDM interface. In this case the user must ensure that no collisions occur on the transmit lines from several SCCs.

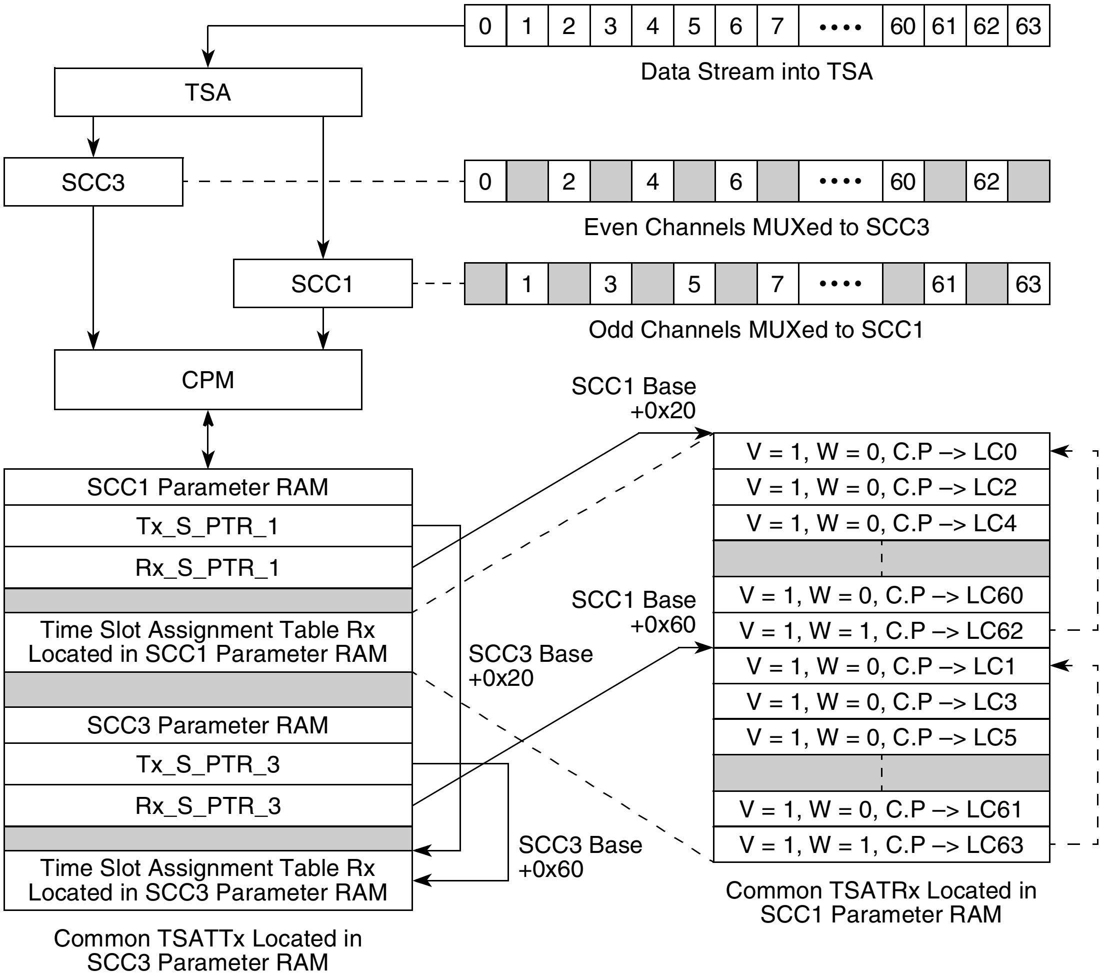


Figure 26-7. Time-Slot Assignment Tables for 64 Channels over 2 SCCs

26.3.4 Channel-Specific Parameters

The channel-specific parameters are located in the lower part of the dual-ported RAM. Each channel occupies 64 bytes of parameters. Physical time slots can be matched to logical channels in several combinations. Unused logical channels leave a hole in the channel-specific parameters that can be used for buffer descriptors for the other SCCs.

The channel-specific area determines the operating mode—HDLC or transparent. Several entries take on different meanings depending on the protocol chosen.

26.3.4.1 Channel-Specific HDLC Parameters

Table 26-4 describes the channel-specific HDLC parameters. Boldfaced parameters must be initialized by the user.

Table 26-4. 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 26-2.
02	**CHAMR**	16	Channel mode register. See Section 26.3.4.1.1, “CHAMR—Channel Mode Register (HDLC).”
04	**TSTATE**	32	Tx internal state —TSTATE defines the internal Tx state. Initialize before enabling the channel. See Section 26.3.4.1.2, “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 26-1. 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 26.3.4.1.3, “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 26-2.

Table 26-4. Channel-Specific HDLC Parameters (continued)

Offset	Name	Width (Bits)	Description
22	**MFLR**	16	Maximum frame length register (host-initialized)—Defines the longest expectable frame for this channel. Its maximum value is 64 Kbytes. The remainder of a frame which is larger than MFLR is discarded and a flag in the last frame's BD is set (LG). An interrupt request (RXF and RXB) might be generated depending on the interrupt mask. The frame length is considered to be everything between flags, including CRC. MFLR is checked every long word, but the content may be on any number of bytes. If MFLR is set to 5 for example, checking is done when 8 bytes have been received. At this point, the SDMA transfers the long word to memory, and all 8 bytes will be in the receive buffer. Also at this point the MFLR violation (>5) is detected and the interrupt may be generated. No more data will be written into this buffer when the MFLR violation is detected.
24	**RSTATE**	32	Rx internal state. See Section 26.3.4.1.4, "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 26-1. MCBASE + RBPTR gives the address for the BD in use.
2E	—	16	Rx internal byte count—Per Channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx Temp) Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0xFFFF_FFFF.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x80FF_FFE0 in HDLC mode prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of zero deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, hold the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel. More information is given in the application note section.
38	RCRC	32	Temp receive CRC—Temp value of CRC calculation result
3C	MAX_cnt	16	Max_length counter—Count length remaining
3E	TMP_MB	16	Temp—Holds MIN(MAX_cnt, Rx internal byte count)

26.3.4.1.1 CHAMR—Channel Mode Register (HDLC)

The channel mode register is a word-length, host-initialized register. Figure 26-8 shows the channel mode register for HDLC operation.

	0	1	2	3	4 6	7	8	9	10 11	12 15
Field	MODE	0	IDLM	ENT	Reserved	POL	CRC	0	Reserved	NOF
Reset	0000_0000_0000_0000									

Figure 26-8. CHAMR—Channel Mode Register (HDLC)

Table 26-5 describes the channel mode register's fields for HDLC operation. Boldfaced parameters must be initialized by the user.

Table 26-5. CHAMR Field Descriptions (HDLC)

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	—	0
2	**IDLM**	Idle mode 0 Idle mode is disabled. No idle patterns are transmitted between frames. After transmitting the NOF + 1 flags, the transmitter starts the data of the frame. If between frames and the frame buffer is not ready, the transmitter sends flags until it can start transmitting the data. The NOF shall be greater or equal to the PAD setting; see Section 26.6.2, "Transmit Buffer Descriptor." If NOF = 0, this is identical to flag sharing in HDLC mode. For a high CPM load or with long bus latencies, the QMC protocol may insert additional flags. 1 Idle mode enabled. At least one idle pattern is transmitted between adjacent frames. If between frames and the frame buffer is not ready, the transmitter sends idle characters. When data is ready, the NOF + 1 flags are sent followed by the data frame. If in IDLE mode and NOF = 1, the following sequence is transmitted: ......init value, FF, FF, flag, flag, data,...... The init value before the idle will be 1's, in this case it is assumed the transmitter was uninitialized. An uninitialized SCC transmits 1s in every position.
3	**ENT**	Enable transmit 0 Disable transmitter. If this bit is cleared and the channel's transmitter is routed to a certain time slot (within TSATTx, see Figure 26-4) 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 CPM does not check the ready bit (R) in the transmit buffer descriptor. 1 The CPM checks the ready bit (R) in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. This bit should always be set by the software at the beginning of a transmit sequence of one or more frames. This bit is cleared (0) by the RISC processor when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), that is, at the end of a frame or at the end of a 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 CPM; the user should not attempt to clear this bit in software.
8	**CRC**	This bit selects the type of CRC when using the HDLC channel mode. 0 16-bit CCITT-CRC is selected for this channel. 1 32-bit CCITT-CRC is selected.
9	—	0
10–11	—	Reserved
12–15	**NOF**	Number of flags. Defines the minimum number of flags before frames. However, even if NOF = 0, at least one flag is transmitted before the first frame. See the description of the IDLM bit for more information.

26.3.4.1.2 TSTATE—Tx Internal State (HDLC)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type. Figure 26-9 shows the TSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**TC2**	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 26-9. TSTATE—TX Internal State (HDLC)

Table 26-6 describes TSTATE fields.

Table 26-6. TSTATE Field Descriptions (HDLC)

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared

26.3.4.1.3 INTMSK—Interrupt Mask (HDLC)

Each event defined in the interrupt circular queue entry maps directly to a bit in INTMSK as shown in Figure 26-11. There is one mask bit for each event—NID (bit 2), IDL (bit 3), MRF (bit 10), UN (bit 11), RXF (bit 12), BSY (bit 13), TXB (bit 14) and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero. Refer to Section 26.5, “QMC Exceptions,” for more detail.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host prior to operation.

	0	1	2	3	4 … 9	10	11	12	13	14	15
Field	V	W	NID	IDL	Channel Number	MRF	UN	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000										

Figure 26-10. Interrupt Table Entry

	0 1	2 3	4 9	10 15
Field	Reserved	Interrupt Mask	Reserved	Interrupt Mask Bits
Reset	0000_0000_0000_0000			

Figure 26-11. INTMSK (HDLC)

26.3.4.1.4 RSTATE—Rx Internal State (HDLC)

The RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type. Figure 26-12 shows the RSTATE register for HDLC operation.

	0 1	2	3 4	5	6 7
Field	—	**GBL**	**BO**	**TC2**	—
Reset	0000_0000_0000_0000				
R/W	R/W				

Figure 26-12. RSTATE—RX Internal State (HDLC)

Table 26-7 describes RSTATE fields.

Table 26-7. RSTATE Field Descriptions (HDLC)

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared

26.3.4.2 Channel-Specific Transparent Parameters

Table 26-8 describes the channel-specific transparent parameters. Boldfaced parameters must be initialized by the user.

Table 26-8. 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 26-2.
02	**CHAMR**	16	Channel mode register. See Section 26.3.4.2.1, "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 26.3.4.2.2, "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 26-1.. 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 26-16.
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 26-2.
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 26.3.4.2.5, "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 26-2. MCBASE + RBPTR gives the address for the BD in use.
2E		16	Rx internal byte count—Per channel: Number of remaining bytes in buffer
30	RPACK	32	(Rx temp)—Packs 4 bytes to 1 long word before writing to buffer. Should be initialized to 0xFFFF_FFFF.
34	**ZDSTATE**	32	Zero deletion machine state—(Host-initialized to 0x02FF_33E0 in transparent mode prior to operation and after a fatal Rx error (global overrun, busy) before channel initialization.)—Contains the previous state of the zero-deletion state machine. The middle 2 bytes, represented by zeros in the initialization value above, holds the received pattern during reception. A window of 16 bits shows the history of what is received on this logical channel.
38	RES	32	—

Table 26-8. 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 26.3.4.2.4, “TRNSYNC—Transparent Synchronization.”
3E	RES	16	—

26.3.4.2.1 CHAMR—Channel Mode Register (Transparent Mode)

The channel mode register is a word-length, host-initialized register. Figure 26-13 shows the channel mode register for transparent mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	15
Field	MODE	RD	1	ENT	—	SYNC	—	POL	0	0	—		0	
Reset	0000_0000_0000_0000													

Figure 26-13. CHAMR—Channel Mode Register (Transparent Mode)

Table 26-9 describes the channel mode register’s fields for transparent operation. Boldfaced parameters must be initialized by the user.

Table 26-9. 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 will not be reversed, transmitting/receiving the LSB of each octet first. 1 The bit order as seen on the channels is reversed, transmitting/receiving the MSB of each octet first.
2	—	1
3	**ENT**	Enable transmit 0 Disable transmitter. If this bit is cleared and the channel’s transmitter is routed to a certain time slot (within TSATTx, see Figure 26-4.) 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.
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 26-9. CHAMR Bit Settings (Transparent Mode) (continued)

Field	Name	Description
7	**POL**	Enable polling. Enables the transmitter to poll the transmit BDs. 0 The CPM will not check the ready (R) bit in the transmit buffer descriptor. 1 The CPM will go check the ready (R) bit in the transmit buffer descriptor. The user can use this bit to prevent unnecessary external bus cycles when checking the ready bit (R) in the buffer descriptor. Software should always set POL at the beginning of a transmit sequence of one or more frames. The RISC processor clears POL (0) when no more buffers are ready in the transmit queue when it finds a buffer descriptor with the R bit cleared (0), that is, at the end of a frame or at the end of a multiframe transmission. To prevent deadlock, software should prepare the new BD, or multiple BDs, and set (1) the ready (R) bit in the BD before setting (1) POL. Note that the CPM automatically clears this bit; the user should never try to clear this bit in software.
8–9	—	0
10–11	—	Reserved
12–15	—	0

26.3.4.2.2 TSTATE—Tx Internal State (Transparent Mode)

TSTATE defines the internal transmitter state. The high byte of TSTATE defines the function code/address type. Figure 26-14 shows the TSTATE register for transparent mode.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**TC2**	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 26-14. TSTATE—TX Internal State (Transparent Mode)

Table 26-10 describes TSTATE fields.

Table 26-10. TSTATE Field Descriptions (Transparent Mode)

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared

26.3.4.2.3 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 26-16. There is one mask bit for each event—UN (bit 11), BSY (bit 13), TXB (bit 14) and RXB (bit 15). Bits that do not map to an event are reserved. Reserved bits must be set to zero.

- 0 = No interrupt request is generated and no new entry is written in the circular interrupt table.
- 1 = Interrupts are enabled.

This register is initialized by the host before operation.

	0	1	2 4	5 9	10	11	12	13	14	15
Field	V	W	—	Channel Number	—	UN	—	BSY	TXB	RXB
Reset	0000_0000_0000_0000									

Figure 26-15. Interrupt Table Entry

	0 10	11 15
Field	Reserved	Interrupt Mask Bits
Reset	0000_0000_0000_0000	

Figure 26-16. INTMSK (Transparent Mode)

26.3.4.2.4 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 (0).

When sending a transparent message over several time slots, it is necessary to know in which time slot the first byte of data appears.

The TRNSYNC word-length register is divided into two parts with the high byte controlling the first received time slot and the low byte controlling the transmitter synchronization.

Take the example of a superchannel of several time slots:

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

> Note that TS*n* is not necessarily the first time slot in the frame. For example, if a superchannel is produced from TS2, TS4, and TS6, the message may be arranged with TS4 holding the first byte, then TS6, and the final byte held in TS2 of the following frame.

The following nine cases in Figure 26-17, named C1 to C9, show different scenarios ranging from a single time slot per logical channel to a superchannel using several time slots. In this application, 24 time slots are routed to this SCC from the SI RAM. After time slot 23, the frame starts with 0 again. The arrow in all the figures illustrates the starting position.

C1 is for a single byte in TS7, so 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) × 2 = 40

C8 is a 4-byte pattern TS8, TS9, TS19, TS20 and TS23, so TSn = 8

Rx Byte: (8 + 1) × 2 = 18

As x = 5, TSn + x = 23, so

TX Byte: (23 + 1) × 2 = 0

C9 is a 4-byte pattern TS19, TS20, TS23, TS8 and TS9, so TSn = 19

Rx Byte: (19 + 1) × 2 = 40

As x = 5, TSn + x = 9, so

TX Byte: (9 + 1) × 2 = 20

NOTE

Case C1 and C2 can be used as described above. A more elegant solution for single time-slot applications is to have the SYNC bit cleared (0) in the channel mode register. Both scenarios produce the same result.

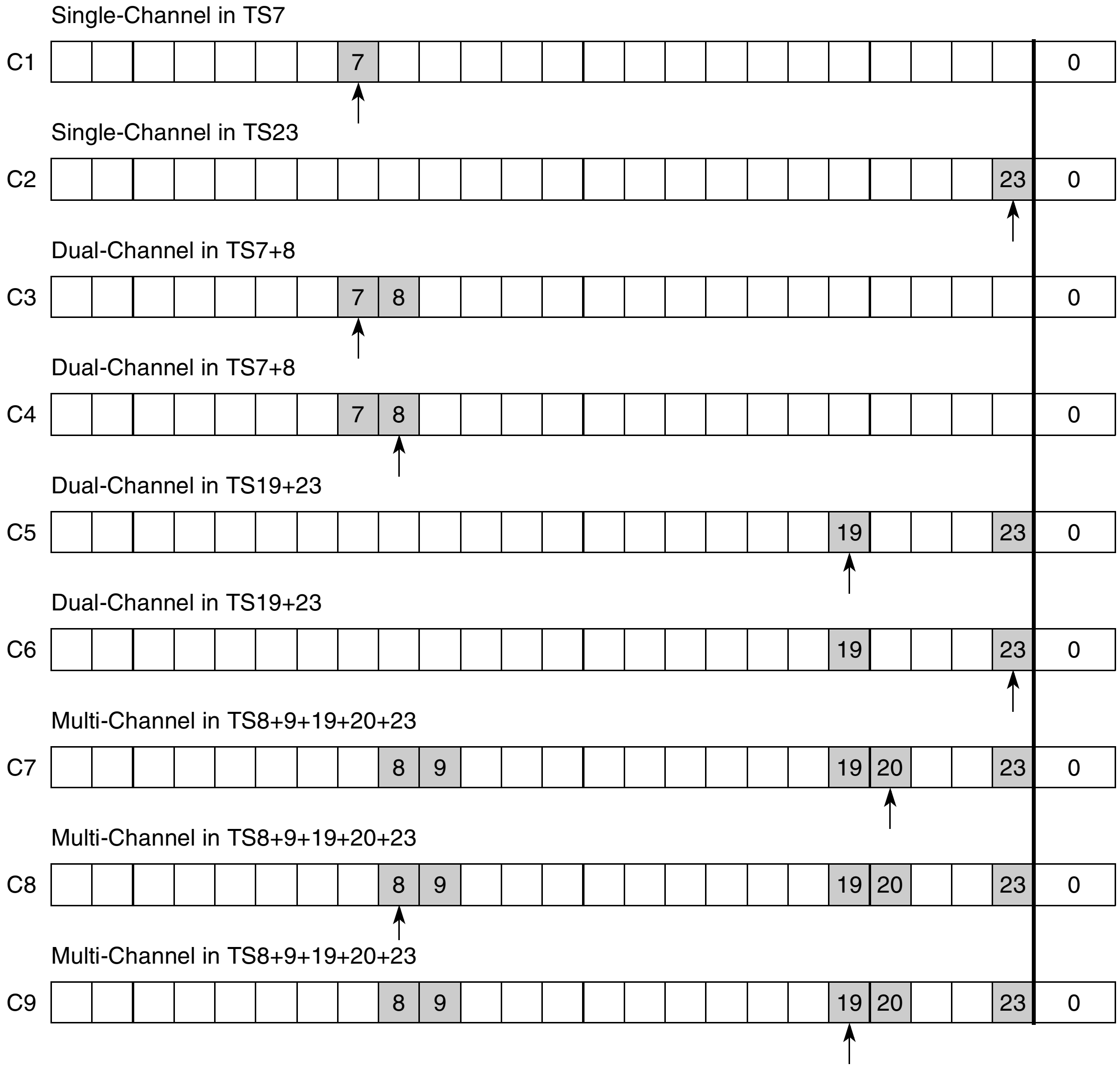


Figure 26-17. Examples of Different T1 Time Slot Allocation

26.3.4.2.5 RSTATE—Rx Internal State (Transparent Mode)

The RSTATE is host-initialized before enabling the channel or after a fatal error (that is, global overrun, busy) or after a STOP Rx command.

The high byte of RSTATE defines the function code/address type. Figure 26-18 shows the RSTATE register for HDLC operation.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	—	
Reset	0000_0000_0000_0000							
R/W	R/W							

Figure 26-18. RSTATE—RX Internal State (Transparent)

Table 26-11 describes RSTATE fields.

Table 26-11. RSTATE Field Descriptions (Transparent)

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 00 Reserved 01 Munged little endian 1x Big endian or true little endian
5	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6–7	—	Reserved, should be cleared.

26.4 QMC Commands

The host issues commands to the QMC by writing to the command register (CPCR). Refer to Section 13.4.1, "CP Command Register (CPCR)."

26.4.1 Transmit Commands

STOP TRANSMIT <channel>

The stop transmit command disables the transmission of data on the selected channel and clears CHAMR[POL]. Upon asserting this command in the middle of a frame, the RISC processor sends an ABORT indication (7F) followed by IDLEs or FLAGs, depending on the mode, on the selected channel. If this command is issued between frames, the RISC processor continues sending IDLEs or FLAGs (depending on CHAMR[IDLM]) in this channel.

The Tx buffer descriptor pointer (TBPTR) is not advanced to the next buffer; see Table 26-4 and Section 26.3.2.8, "Global Multichannel Parameters."

Set the CHAMR[POL] bit to 1 to start transmission or to continue after a stop command.

Only after transmission starts for a deactivated channel, which is identified by a cleared V bit in the time slot assignment table or a cleared ENT bit, is it necessary to initialize ZISTATE and TSTATE before setting CHAMR[POL].

To deactivate a channel, clear both the V bit in the TSA table and CHAMR[ENT].

26.4.2 Receive Commands

STOP RECEIVE<channel>

The stop receive command forces the receiver of the selected channel to stop receiving. After issuing this command, the microcode does not change any of the receive parameters in the dual-ported RAM. The command immediately stops activity on the channel. It does not wait for a frame reception to be finished. Because the current buffer descriptor can remain open if a reception was in progress, it is possible to get errors on this buffer once you restart reception. When restarting, set ZDSTATE first and RSTATE afterwards.

Initialize ZDSTATE and RSTATE to their initial values to start reception or to continue receiving after a stop command.

NOTE

No commands exist to start transmission and reception. This function is realized through the GSMR register.

26.5 QMC Exceptions

QMC interrupt handling involves two principle data structures—the SCC event register (SCCE) and the circular interrupt table. Figure 26-19 illustrates the circular interrupt table.

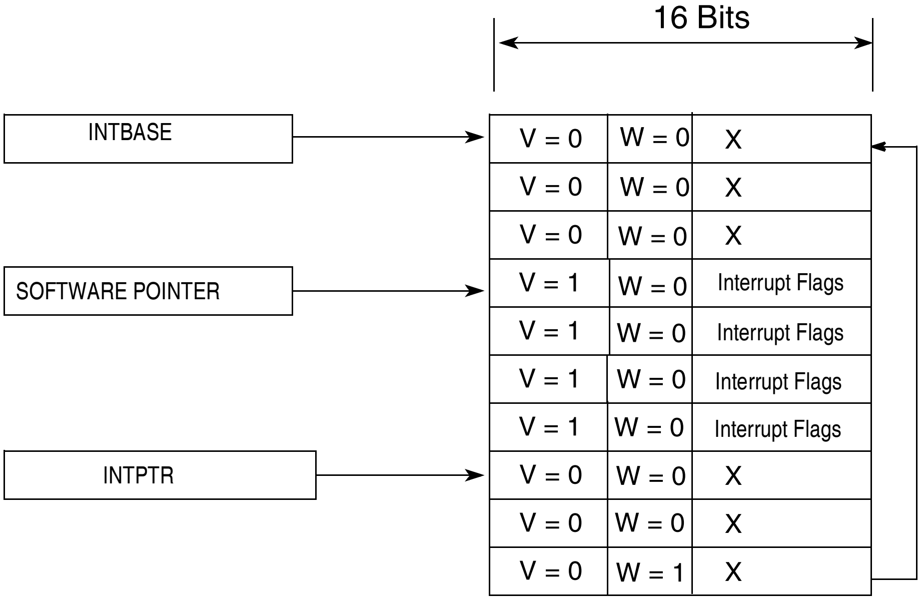


Figure 26-19. Circular Interrupt Table in External Memory

INTBASE (interrupt base) points to the starting location of the queue in external memory, and INTPTR (interrupt pointer) marks the current empty position available to the RISC processor. Both pointers are host-initialized global QMC parameters; see Table 26-1. The entry whose W (wrap) bit is set to 1 marks the end of the queue. When one of the QMC channels generates an interrupt request, the RISC processor writes a new entry to the queue. In addition to the channel's number, this entry contains a description of the exception. The V (valid) bit is then set and INTPTR is incremented. When INTPTR reaches the entry with W = 1, INTPTR is reset to INTBASE.

An interrupt is written to the interrupt table only if it survives a mask with the INTMASK (interrupt mask) register. Following a write to the queue, the QMC protocol updates the SCC event register (SCCE) according to the type of exception.

Following a request that is not masked out by the INTMASK or the SCCM (SCC mask) register, an interrupt is generated to the host. The host reads the SCCE to determine the cause of interrupt. A dedicated SCCE bit (GINT) indicates that at least one new entry was added to the queue. After clearing GINT, the host starts processing the queue. The host then clears this entry’s valid bit (V). The host follows this procedure until it reaches an entry with V = 0, indicating an invalid entry.

NOTE

It is very important that the user clear all bits but the wrap bit in a queue entry even though its valid bit may be cleared. Clearing only the interrupt bits confuse user software with incorrect channel interrupts.

26.5.1 Global Error Events

A global error affects the operation of the SCC. A global error can occur for two reasons— serial data rates being too high for the CPM to handle, and CPM bus latency being too long for correct FIFO operation.

There are two global errors— global transmitting underrun (GUN) and global receiver overrun (GOV). GUN indicates that transmission has failed due to lack of data; and GOV indicates that the receiver has failed because the RISC processor did not write previous data to the receive buffer. In both cases, it is unknown which channel(s) are affected.

Nonglobal, individual channel errors are handled differently. See Section 26.5.3, “Interrupt Table Entry,” for underrun and overrun in a specific channel.

The incoming data to the CPM is governed by transfers between the SCC and the SI. Every transfer in either direction causes a request to the CPM state machine. If requests are received too quickly, the CPM may lose track of the mapping of serial data to the QMC channels. If this happens, the CPM has to cause a global error to halt activity on all QMC channels until user software intervenes. Note that this problem typically indicates a performance-related design problem. Also note that latency is very important.

The other error condition is bus latency. A receiving channel submits data to the FIFO for transfer to external memory as long as the channel operates normally. If the bus latency for the SDMA channels is too long and the receive FIFO is filled and overwritten, a receiver overflow occurs. The overwriting channels cannot be traced, affecting entire QMC operation.

A similar situation can occur during transmission when the SDMA cannot fill the FIFO from external memory because of bus latency. Again, it cannot be determined which channel is underrun, and the whole QMC operation is affected.

Global errors are unlikely to occur in normal system operation, if correct serial speed is used. The only area of concern is data movement between the FIFO and external memory. To avoid problems, the user must understand the bus arbitration mechanism of the QUICC and meet the latency requirements.

26.5.1.1 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 SCCE register.
- Stops reading data from buffer.
- Sends IDLEs or FLAGs in all time slots depending on channel mode settings until the host does the following:

  Host initializes all transmitting channels and time slots by preparing all buffer descriptors for transmission (R bits are set) and setting the POL bit. No other re-initialization is needed.

26.5.1.2 Global Overrun (GOV) in the FIFO

A global overrun affects all channels operating from an SCC. Following GOV, the QMC performs the following:

- Updates the RSTATE register to prevent further reception on this channel. Bit 20 in the RSTATE register indicates that the receiver is stopped.
- Generates an interrupt request to the host (if enabled) and sets the GOV bit in the SCCE.
- Stops writing data to all channels' buffers.
- Waits for host to initialize all the receiving channels by setting first the ZDSTATE followed by the RSTATE to their initial values.

26.5.1.3 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 26.7, "QMC Initialization."

26.5.2 SCC Event Register (SCCE)

The QMC's SCCE is a word-length register used to report events and generate interrupt requests. See Figure 26-20 and Table 26-12 for SCCE field descriptions. For each of its flags, a corresponding programmable mask/enable bit in the SCCM determines whether an interrupt request is generated. If a bit in the SCCM register is zero, the corresponding interrupt flag does not survive, and the CPM does not proceed with its usual interrupt handling. If a bit in the SCCM is set, the corresponding interrupt flag in the SCCE survives, and the SCC event bit is set in the CPM interrupt-pending register. See Figure 26-21 for SCCM assignments.

	0 3	4	5	6	7
Field	—	IQOV	GINT	GUN	GOV

Figure 26-20. SCC Event Register

Table 26-12. SCC Event Register Field Descriptions

Field	Name	Description
0–3	—	Reserved
4	IQOV	Interrupt table (interrupt queue) overflow 0 No interrupt table overflow has occurred. 1 An overflow condition in the circular interrupt table occurs (and an interrupt request is generated). This condition occurs if the RISC processor attempts to write a new interrupt entry into an entry that was not handled by the host. Such an entry is identified by V = 1. This bit should be cleared immediately after reading the SCCE and recognizing this condition by writing a 1 to its location in the SCCE. If this condition occurs, the interrupt last received is lost. It does not overwrite the first entry that is still to be handled by the CPU.
5	GINT	Global interrupt 0 No global interrupt has occurred. 1 This flag indicates that at least one new entry in the circular interrupt table has been generated by the QMC. The host clears GINT by writing a 1 to its location in SCCE. After clearing it, the host reads the next entry from the circular interrupt table, and starts processing a specific channel’s exception. The user must make sure that no more valid interrupts are pending in the interrupt table after clearing the GINT bit, before performing the RTE to avoid deadlock. This procedure ensures that no pending interrupts exist in the queue.
6	GUN	Global transmitter underrun 0 No global transmitter underrun has occurred. 1 This flag indicates that an underrun occurred in the SCC’s transmitter FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GUN bit in the SCCE, the QMC stops transmitting data on all channels. The TDM Tx line goes into idle mode. This error affects only the transmitter; the receiver continues to work. After initializing all the individual channels, the host may resume transmitting. If enabled in the SCCM, an interrupt request is generated when GUN is set. The host may clear GUN by writing 1 to its location in the SCCE.
7	GOV	Global receiver overrun 0 No global receiver overrun has occurred. 1 This flag indicates that an overrun occurred in the SCC’s receiver FIFO. This error is fatal since it is unknown which channel(s) are affected. Following the assertion of the GOV bit in the SCCE, the QMC stops receiving data on all channels. Data is no longer written to memory. This error affects only the receiver; the transmitter continues to work. After initializing all the individual channels, the host may resume receiving. If enabled in SCCM, an interrupt request is generated when GOV is set. The host may clear GOV by writing 1 to its location in the SCCE.

	0 3	4	5	6	7
Field	—	IQOV	GINT	GUN	GOV

Figure 26-21. SCCM Register

26.5.3 Interrupt Table Entry

The interrupt table contains information about channel-specific events. Its flags are shown in Figure 26-22. Note that some bits have no meaning when operating in transparent mode. For more detailed description on which bits are used in HDLC and transparent operation, refer to Section 26.3.4, “Channel-Specific Parameters.” Table 26-13 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 26-22. Interrupt Table Entry

Table 26-13. Interrupt Table Entry Field Descriptions

Field	Name	Description
0	V	Valid bit 0 Entry is not valid 1 Valid entry containing interrupt information Upon generating a new entry, the RISC processor sets this bit. The V bit is cleared by the host immediately after it reads the interrupt flags in this entry (before processing the interrupt). The V bits in the queue are host-initialized. During the initialization procedure, the host must clear those bits in all queue entries.
1	W	Wrap bit 0 This is not the last entry in the circular interrupt table. 1 This is the last circular interrupt table entry. The next event’s entry is written/read (by RISC/host) from the address contained in INTBASE. During initialization, the host must clear all W bits in the queue except the last one which must be set. The length of the queue is left to the user and can be a maximum of 64 Kbytes.
2	NID	Not idle 0 No NID event has occurred. 1 A pattern which is not an idle pattern was identified. NID interrupts are not generated in transparent mode.
3	IDL	Idle 0 No IDL event has occurred. 1 The channel’s receiver has identified the first occurrence of HDLC idle (FFFE) after any non-idle pattern. IDL interrupts are not generated in transparent mode.
4–9	Channel number	Identifies the requesting logical channel index (0–31).
10	MRF	Maximum receive frame length violation—This interrupt occurs in HDLC mode when more than MFLR number bytes are received. As soon as MFLR is exceeded, this interrupt is generated and the remainder of the frame is discarded. At this point the receive buffer is not closed and the reception process continues. The receive buffer is closed upon detecting a flag. The length field written to this buffer descriptor is the entire number of bytes received between the two flags. MRF interrupts are not generated in transparent mode. **Note**: The MRF interrupt is generated directly when the MFLR value is a multiple of 4 bytes. The checking of this is done on a long-word boundary whenever the SDMA transfers 32 bits to memory. If MFLR is not aligned to 4x bytes, this interrupt may be 1- to 3-byte timings late for this channel. In any case, the violation can be checked to any number of bytes. The last entry in the data buffer is always a full long word.

Table 26-13. Interrupt Table Entry Field Descriptions (continued)

Field	Name	Description
11	UN	Tx no data 0 No UN event has occurred. 1 There is no valid data to send to the transmitter. The transmitter sends an abort indication consisting of 16 consecutive 1's and then sends idles or flags according to the protocol and the channel mode register setting. This error occurs when a transmit channel has no data buffer ready for a multibuffer transmission already in progress. Transmission of a frame is a continuous bit stream without gaps or interruption. When a buffer is not ready in the middle of this sequence, it is an error situation. This can be viewed as channel underrun. The transmitter for this channel is stopped. See Section 26.7.1, "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 26.7.2, "Restarting the Receiver."
14	TXB	Tx buffer 0 No TXB event has occurred. 1 A buffer has been completely transmitted. This bit is set (and an interrupt request is generated) as soon as the programmed number of PAD characters (or the closing flag, for PAD = 0) is written to the SCC's transmit FIFO. The number of PAD characters determines the timing of the TXB interrupt in relation to the closing flag sent out at TXD. See Section 26.6, "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.

26.5.4 Interrupt Handling

To handle interrupts by the QMC, first read the SCCE register. Write back immediately all the bits that you recognize and handle. This clears the respective interrupt events. It is, for example, incorrect to first handle all the new interrupt table entries and clear GINT afterwards. To avoid deadlocks in software, clear the recognized interrupt bits in the SCCE before actually handling the interrupt entries. Clear only the bits being handled. Never clear bits for events in the SCCE 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 will confuse the software when the entry is used again. QMC will only OR in new bits and numbers. It will not overwrite and clear bits that were set previously.

26.5.5 Channel Interrupt Processing Flow

Figure 26-23 illustrates the flow of a channel interrupt. Note that this does not describe the processing of the global interrupts GUN and GOV.

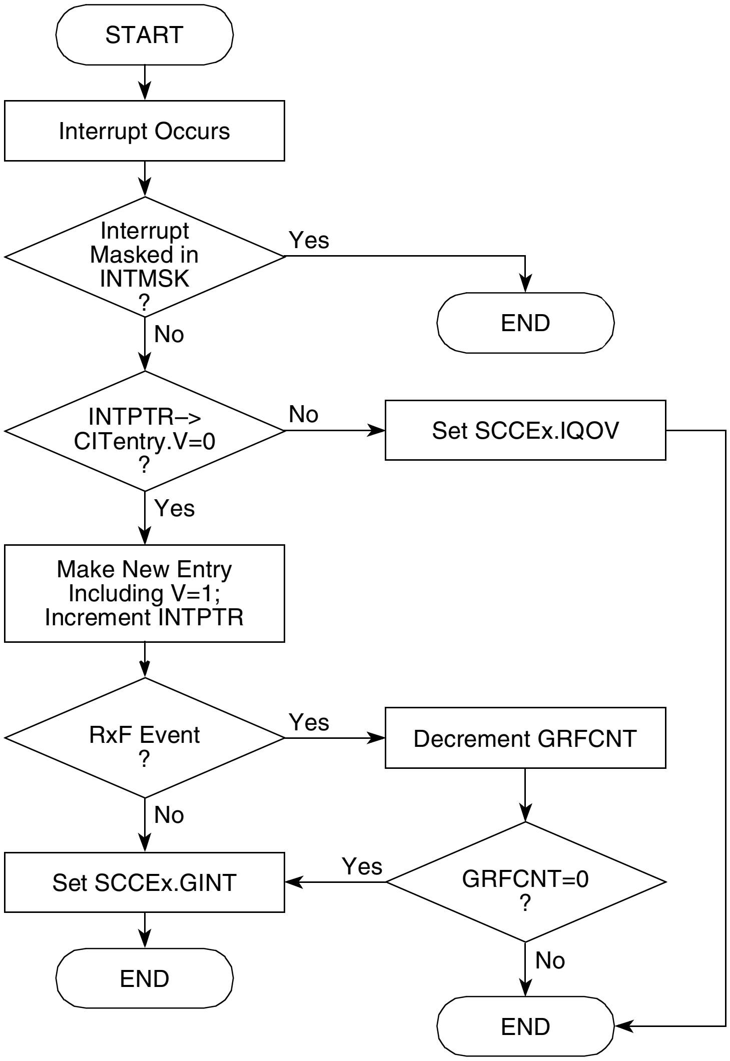


Figure 26-23. Channel Interrupt Flow

26.6 Buffer Descriptors

QMC buffer descriptors are located within 64 Kbytes in external memory; see Figure 26-2. Each buffer descriptor contains key information about the buffer it defines.

The first two sections describe the contents of the receive and transmit buffer descriptors for the QMC protocol.

The third section discusses the placement of QMC and non-QMC buffer descriptors in internal and external memory.

NOTE

> The CPM ORs the various status bits into the BD. Clear all the status bits generated by the CPM before (re-)enabling the BD, or you may confuse your own software with leftover old status values.

26.6.1 Receive Buffer Descriptor

Figure 26-24 shows a receive buffer descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	—	**W**	**I**	L	F	**CM**	—	UB	—	LG	NO	AB	CR	—	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**Rx DATA BUFFER POINTER**															
OFFSET + 6																

Notes: Entries in boldface must be initialized by the user.

Figure 26-24. Receive Buffer Descriptor (RxBD)

Table 26-14 describes the individual fields of a receive buffer descriptor. Boldfaced entries must be initialized by the user.

Table 26-14. RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The data buffer associated with this BD has been filled with received data, or data reception has been aborted due to an error condition. The user is free to examine or write to any fields of this RxBD. The CP does not use this BD again while the empty bit remains zero. 1 The data buffer associated with this BD is empty, or reception is in progress. This RxBD and its associated receive buffer are in use by the CP. When E = 1, the user should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table. After this buffer has been used, the CP receives incoming data into the first BD in the table (the BD pointed to by RBASE). The number of RxBDs in this table is programmable and is determined by the wrap bit.
3	**I**	Interrupt 0 The RXB bit is not set after this buffer has been used, but RXF operation remains unaffected. 1 The RXB or RXF bit in the HDLC interrupt circular table entry is set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).

Table 26-14. RxBD Field Descriptions (continued)

Bits	Name	Description
4	L	Last in frame (only for HDLC mode of operation). The HDLC controller sets L = 1, when this buffer is the last in a frame. This implies the reception either of a closing flag or of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 This buffer is not the last in a frame. 1 This buffer is the last in a frame.
5	F	First in frame. The HDLC controller sets F = 1 for the first buffer in a frame. In transparent mode, F indicates that there was a synchronization before receiving data in this BD. 0 This is not the first buffer in a frame. 1 This is the first buffer in a frame.
6	**CM**	Continuous mode 0 Normal operation. (The empty bit (bit 0) is cleared by the CP after this BD is closed.) 1 The empty bit (bit 0) is not cleared by the CP after this BD is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD. However, if an error occurs during reception, the empty bit is cleared regardless of the CM bit setting.
7	—	Reserved, should be cleared
8	**UB**	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9	—	Reserved, should be cleared
10	LG	Rx frame length violation (HDLC mode only). Indicates that a frame length greater than the maximum value was received in this channel. Only the maximum-allowed number of bytes, MFLR rounded to the nearest higher word alignment, are written to the data buffer. This event is recognized as soon as the MFLR value is exceeded when data is word-aligned. When data is not word-aligned, this interrupt occurs when the SDMA writes 64 bits to memory. The worst-case latency from MFLR violation until detected is 7 bytes timing for this channel. When MFLR violation is detected, the receiver is still receiving even though the data is discarded. The buffer is closed upon detecting a flag, and this is considered to be the closing flag for this buffer. At this point, LG is set (1) and an interrupt may be generated. The length field for this buffer is everything between the opening flag and this last identifying flag.
11	NO	Rx nonoctet-aligned frame. A frame of bits not divisible exactly by eight was received. NO = 1 for any type of nonalignment regardless of frame length. The shortest frame that can be detected is of type FLAG-BIT-FLAG, which causes the buffer to be closed with NO error indicated. The Non Octet byte is not written into memory and the data length DL field is not updated with this count.
12	AB	Rx abort sequence. A minimum of seven consecutive 1s was received during frame reception. Abort is not detected between frames. The sequence Closing-Flag, data, CRC, AB, data, opening-flag... does not cause an abort error. If the abort is long enough to be an idle, an idle line interrupt may be generated. An abort within the frame is not reported by a unique interrupt but rather with a RXF interrupt and the user has to examine the BD.
13	CR	Rx CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14	—	Reserved, should be cleared
15	—	Reserved, should be cleared

NOTE: RxBD and MRBLR

When the length of a received frame is an exact multiple of MRBLR, it is possible that the CP will not mark the BD that contains the last of the frame's data as the last BD in the frame. The CP will close the BD, mark it with E=0, set the Data Length field to be the length of the data in this buffer, but leave L (bit 4) cleared. Then, the CP will close the next BD, erroneously mark it as containing data (E=0), and mark it as the last BD in the frame (L=1).

Figure 26-25 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.

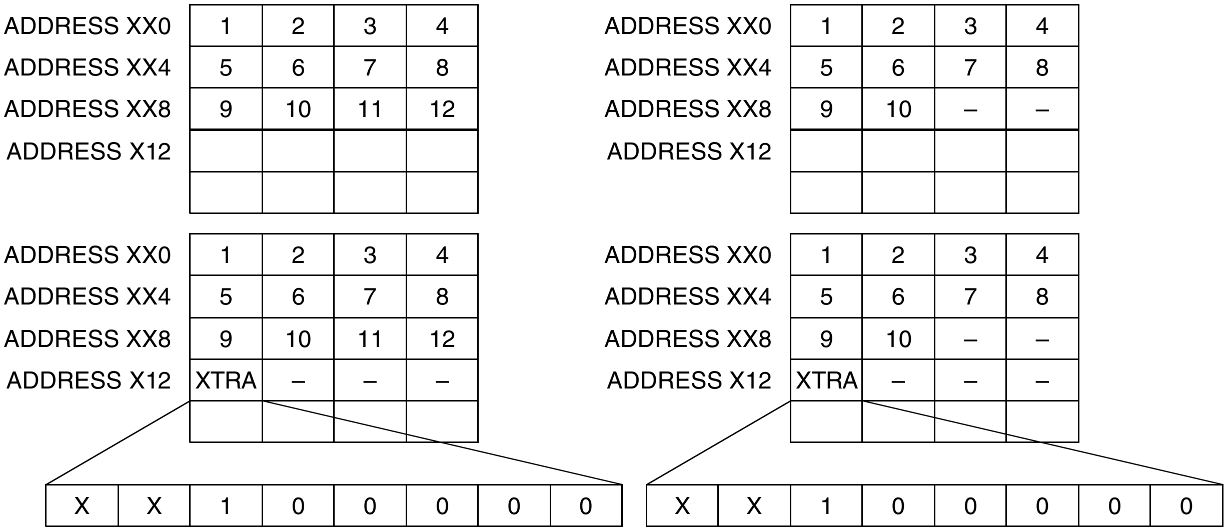


Figure 26-25. Nonoctet Alignment Data

In the bottom case, two more bits are received. The frame length is now 13 bytes, and the address positions X13 through X15 point to invalid data. Address position X12 contains information about the 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 26-25 is written as 32-bit word. Under special, but quite possible circumstances the XTRA data is written four bytes further than indicated. Therefore, users must allocate MRBLR+8 bytes for each buffer area for QMC to avoid the risk of these memory areas being overwritten with XTRA info.

26.6.2 Transmit Buffer Descriptor

Figure 26-26 shows the transmit buffer descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	R	—	W	I	L	TC	CM	—	UB	—	—	—	PAD			
OFFSET + 2	DATA LENGTH															
OFFSET + 4	Tx DATA BUFFER POINTER															
OFFSET + 6																

Notes: Entries in boldface must be initialized by the user.

Figure 26-26. Transmit Buffer Descriptor (TxBD)

Table 26-15 describes the individual fields of a transmit buffer descriptor. Boldfaced entries must be initialized by the user.

Table 26-15. 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 CPM clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is being transmitted. If R = 1, the user cannot write to fields of this buffer descriptor.
1	**—**	—
2	**W**	Wrap (final buffer descriptor in table) 0 This is not the last buffer descriptor in the TxBD table. 1 This is the last descriptor in the Tx buffer descriptor table. After this buffer is used, the CPM transmits data from the first buffer descriptor in the table (the buffer descriptor pointed to by TBASE). The number of TxBDs in this table is programmable and is determined only by the wrap bit and the overall space constraints of the dual-ported RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the circular interrupt table entry is set when the controller services this buffer. This bit can cause an interrupt (if enabled).
4	**L**	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
5	**TC**	Tx CRC (HDLC mode only). This bit is valid only when L = 1; otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send an erroneous CRC after the data. 1 Transmit the CRC sequence after the last data byte.

Table 26-15. 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 CPM after this buffer descriptor is closed, allowing the associated data buffer to be retransmitted automatically when the CPM next accesses this buffer descriptor.
7	—	—
8	**UB**	User bit. The CPM never touches, sets, or clears this user-defined bit. The user determines how this bit is used. For example, it can be used to signal between higher-level protocols whether a buffer has been processed by the CPU.
9–11	—	—
12–15	**PAD**	Padding bits. These four bits indicate the number of PAD characters (0x7E or 0xFF depending on IDLM mode in the CHAMR register) that the transmitter sends after the closing flag. The transmitter issues a TXB interrupt only after sending the programmed value of pads to the Tx FIFO. The user can use the PAD value to guarantee that a TXB interrupt occurs after the closing flag has been sent on the TXD line. PAD = 0 means the TXB interrupt is issued immediately after sending the closing flag to the Tx FIFO. The number of PAD characters depends on the FIFO size and the number of time slots in use. An example explains the calculation: In SCC1 the FIFO is 32 bytes. If 16 time slots are used in the link, the resulting number of PAD characters is 32/16 = 2, to append to this buffer to ensure that the TXB interrupt is not given before the closing flag has been transmitted through the TXD line. The number of PAD characters must not exceed the NOF characters, ensuring that the closing of one buffer (the interrupt generation) occurs before the start of the next frame (clearing of R-bit). After the sequence of a closing flag followed by (PAD + 1) flag characters, a TXB interrupt will be generated; see Figure 26-27.
16–31	**DL**	Data length. The data length is the number of bytes the CPM should transmit from this buffer descriptor's data buffer. It is never modified by the CPM. This field should be greater than zero.
32–63	**TxBP**	Tx buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CPM.

Figure 26-27 shows a TXB interrupt generated after (PAD + 1) flag characters following the closing flag. Four flags (NOF = 3) precede the next data. To set up this sequence correctly, the PAD value must not exceed NOF.

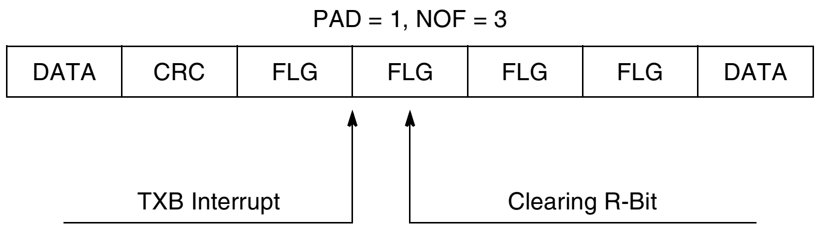


Figure 26-27. Relation between PAD and NOF

26.6.3 Placement of Buffer Descriptors

The internal dual-ported RAM is used to store the buffer descriptors for all non-QMC operation. This solution causes minimum loading of the external bus. When starting any operation or switching between buffers during operations, several accesses must be made by the CPM to find the actual data buffers and to read and write control and status information. This process is unseen by the user for internal accesses, and any external bus master or memory refresh control can occur uninterrupted.

26.6.3.1 Parameter RAM Usage for QMC over Several SCCs

There are two possible memory configurations for operating the QMC protocol on several SCCs. For each SCC in QMC protocol, a global parameter area is used, consuming at most 190 bytes if every global area contains the routing tables. The time-slot assignment tables (TSATRx and TSATTx) together occupy 128 bytes. The tables for each SCC in the parameter RAM can be duplicated, requiring 190 bytes per SCC.

Alternatively, multiple SCCs can use one set of common time slot assignment tables (TSATRx and TSATTx), as described in Section 26.3.2.1, “TSATRx/TSATTx Pointers and Time-Slot Assignment Table.” One SCC RAM page uses 190 bytes, 62 bytes for parameter fields and 128 bytes for the common time slot assignment tables, allowing the other SCCs to use only 62 bytes of parameter RAM for QMC protocol, freeing their 128 bytes of time-slot assignment table space.

26.6.3.2 Internal Memory Structure

For details about the MPC8272’s internal memory, refer to Chapter 3, “Memory Map,” and Section 13.5, “Dual-Port RAM.”

To support 32 channels, only 2-Kbyte dual-ported RAM is needed for channel-specific parameters. Each logical channel occupies 64 bytes; thus 32 channels require 2 Kbytes, leaving 2 Kbytes free in the dual-ported RAM for buffer descriptors for other protocols.

To support 64 channels, 4-Kbyte dual-ported RAM is required for channel-specific parameters. Each logical channel occupies 64 bytes, requiring 4 Kbytes for 64 channels.

Non-QMC protocol implementations may be constrained by these memory requirements since their buffer descriptors need to use internal memory space.

If fewer than 64 logical channels are used or if multiple time slots are concatenated to form super channels, using one QMC channel to manage those time slots, space is freed in the dual-ported RAM. Each unused logical channel creates a 64-byte hole in the dual-ported RAM. This area is free for buffer descriptors for any SCC. QMC channels can also use this space instead of external memory for buffer descriptors, reducing the load on the external bus.

26.7 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 CPCR[OPCODE] (see Section 13.4.1, “CP Command

Register (CPCR)") and GSMR_L[MODE] (see Section 19.1.1, "The General SCC Mode Registers (GSMR1–GSMR4)") set for QMC operation.

Step 1. Initialize QMC global parameters, including all parameter fields, Tx and Rx time-slot assignment tables and framer tables.

1. Initialize channel-specific parameters.
2. Initialize TxBDs and corresponding Tx data buffers
3. Initialize RxBDs
4. Connect the SCC to the TDM
5. interface through the SCC clock muxing register CMXSCR
6. Initialize SCC registers, including enabling the SCC
7. Program RX and TX SIRAM to point appropriate time slots to the SCC used for QMC.
8. Initialize SI registers
9. Enable TDM

26.7.1 Restarting the Transmitter

A global underrun may require the SCC transmitter to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

- Prepare buffer descriptors.
- 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.

26.7.2 Restarting the Receiver

A global receiver overrun may require the SCC receiver to be restarted. However, for channel-specific errors, only the affected channel need be restarted. The following steps are required to restart each channel:

- Prepare buffer descriptors.
- Renitialize the ZDSTATE.
- Reinitialize the RSTATE.

26.7.3 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 26.4.1, "Transmit Commands." The transmitter will continue to send IDLEs or FLAGs according to the channel mode register setting. To deactivate a channel, the V bit has to be cleared in the time-slot assignment table and the ENT bit has to be cleared in the channel mode register.

To stop a channel while receiving, use the STOP command as described in Section 26.4.2, "Receive Commands," and perform a restart as described above.

Chapter 27
Universal Serial Bus Controller

The universal serial bus (USB) controller allows the MPC8272 to communicate with other devices via a USB connection. This chapter describes the MPC8272's USB controller, including basic operation, the parameter RAM, and registers. It also provides programming examples for initializing host mode and function mode of the USB controller.

27.1 USB Integration in the MPC8272

The following restrictions apply when enabling the USB controller in the MPC8272:

- The USB controller pins are multiplexed with SCC3 pins in the parallel I/O. Refer to Chapter 37, "Parallel I/O Ports." The user programs the parallel I/O registers as if SCC3 was being used. Therefore, if the USB controller is enabled, SCC3 cannot be used in NMSI mode. SCC3 may be used with the TSA by setting CMXSCR[SC3].
- The user must program CMXSCR[TS3CS] (refer to Section 15.4.5, "CMX SMC Clock Route Register (CMXSMR)") to the desired source for USB when the USB controller is enabled.

27.2 Overview

The universal serial bus (USB) is an industry-standard extension to the PC architecture. The USB controller on the MPC8272 supports data exchange between a wide range of simultaneously accessible peripherals. Attached peripherals share USB bandwidth through a host-scheduled, token-based protocol.

The USB physical interconnect is a tiered-star topology, and the center of each star is a hub. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or a function. The USB transfers signal and power over a four-wire cable, and the signaling occurs over two wires and point-to-point segments. The USB full speed signaling bit rate is 12 Mbps. Also, a limited capability low-speed signaling mode is defined at 1.5 Mbps. Refer to the USB Specification Revision 2.0 for further details. It can be downloaded from http://www.usb.org.

The MPC8272 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 (loop-back diagnostics).

27.2.1 USB Controller Key Features

The USB function mode features are as follows:

- Four independent endpoints support control, bulk, interrupt, and isochronous data transfers

- CRC16 generation and checking
- CRC5 checking
- NRZI encoding/decoding with bit stuffing
- 12- or 1.5-Mbps data rate
- Flexible data buffers with multiple buffers per frame
- Automatic retransmission upon transmit error

The USB host controller features are as follows:

- Supports control, bulk, interrupt, and isochronous data transfers
- CRC16 generation and checking
- NRZI encoding/decoding with bit stuffing
- Supports both 12- and 1.5-Mbps data rates (automatic generation of preamble token and data rate configuration). Note that low-speed operation requires an external hub.
- Flexible data buffers with multiple buffers per frame
- Supports local loopback mode for diagnostics (12 Mbps only)

27.3 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
- Incrementing the frame number and generating CRC5 for the SOF (Start of Frame) token once per frame (1 ms)

Additionally, when using the packet-level interface described in Section 27.5.1.1, “Packet-Level Interface,” the tokens must be prepared by the software.

Because the MPC8272 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.

27.3.1 USB Controller Pin Functions and Clocking

The USB controller interfaces to the USB bus through a differential line driver and differential line receiver. The $\overline{\text{OE}}$ (output enable) signal enables the line driver when the USB controller transmits on the bus.

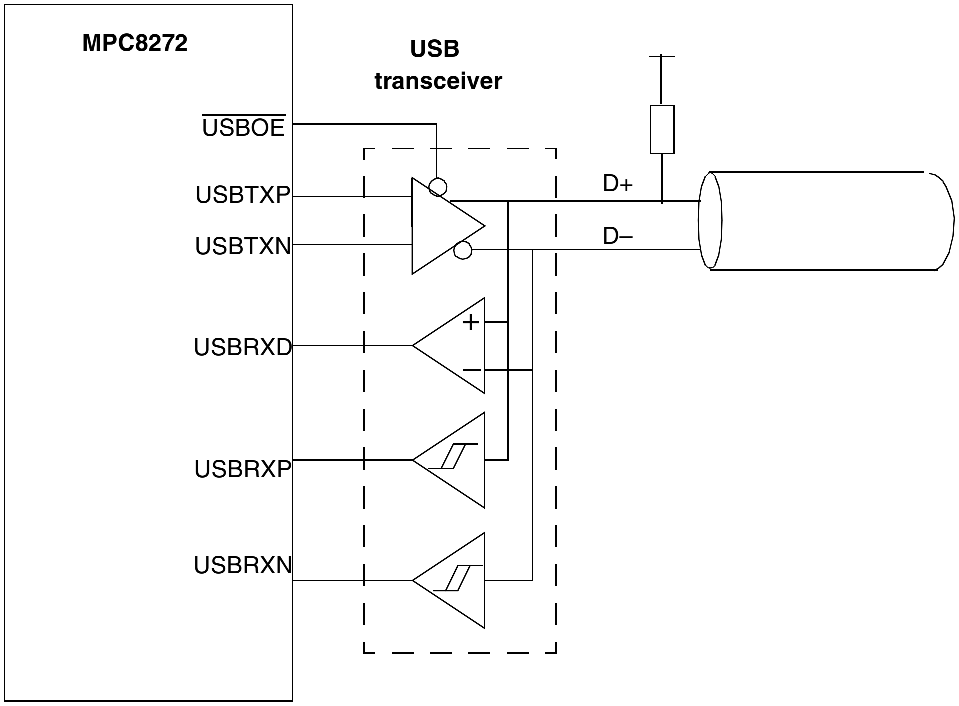


Figure 27-1. USB Interface

The reference clock for the USB controller (USBCLK) is used by the DPLL circuitry to recover the bit rate clock. The source for USBCLK is selected in CMXSCR[TS3CS] (refer to Section 15.4.2, “CMX SI2 Clock Route Register (CMXSI2CR)”). The MPC8272 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 27-1. Additional control lines that might be needed by some transceivers (e.g., speed select, low power control) may be supported by general purpose output lines.

Table 27-1. USB Pins 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 27-1. USB Pins 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>

27.4 USB Function Description

As shown in Figure 27-2, 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.

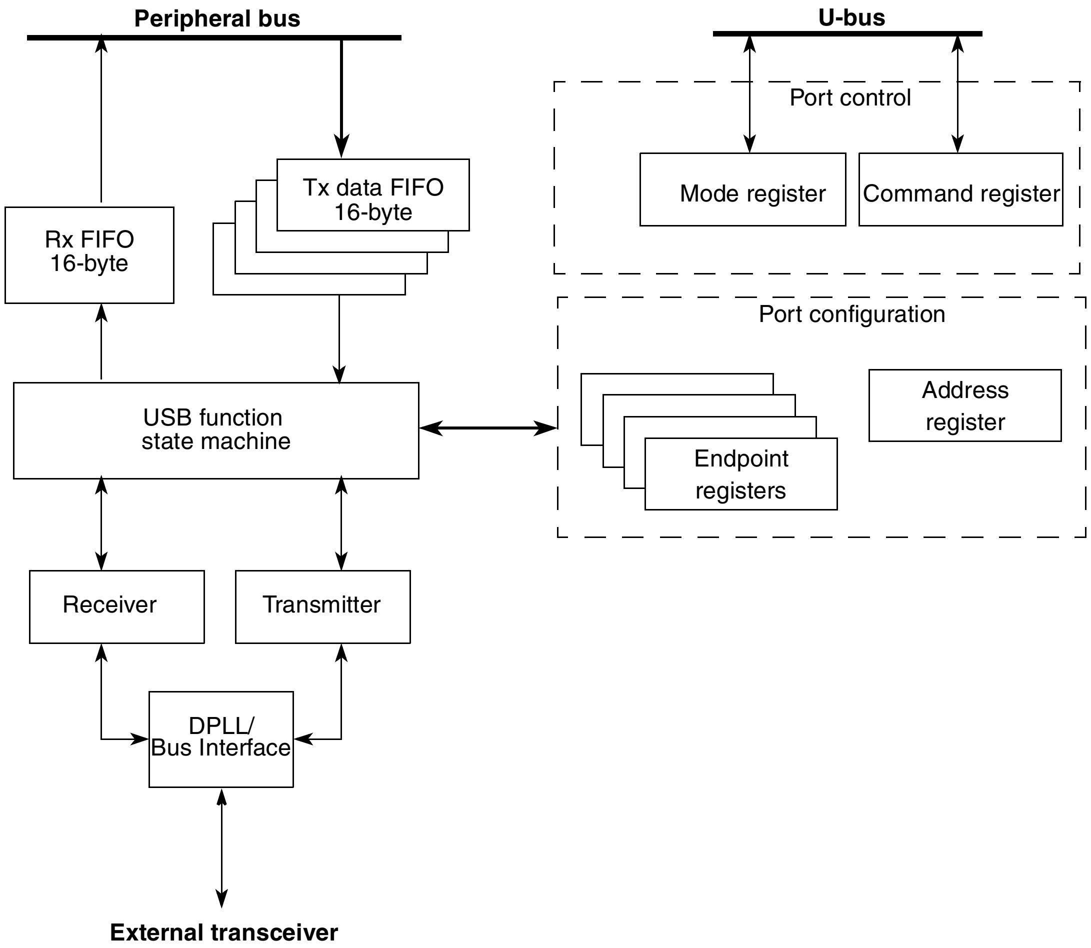


Figure 27-2. USB Function Block Diagram

27.4.1 USB Function Controller Transmit/Receive

After reset condition, the USB function is addressable at the default address (0x00). During the enumeration process the USB function is assigned by the host with a unique address. The USB slave address register (refer to Section 27.5.7.2, “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 27.5.7.3, “USB Endpoint Registers (USEP1–USEP4)”).

NOTE

It is mandatory that endpoint 0 be configured as a control transfer type. This endpoint is used by the USB system software as a control pipe. Additional control pipes may be provided by other endpoints.

Once enabled, the USB function controller looks for valid token packets. Figure 27-3 and Table 27-2 describe the behavior of the USB controller for each token. Tokens that are not valid (that is, PID check fails or CRC check fails or packet length is not 3 bytes) are ignored by the USB function controller.

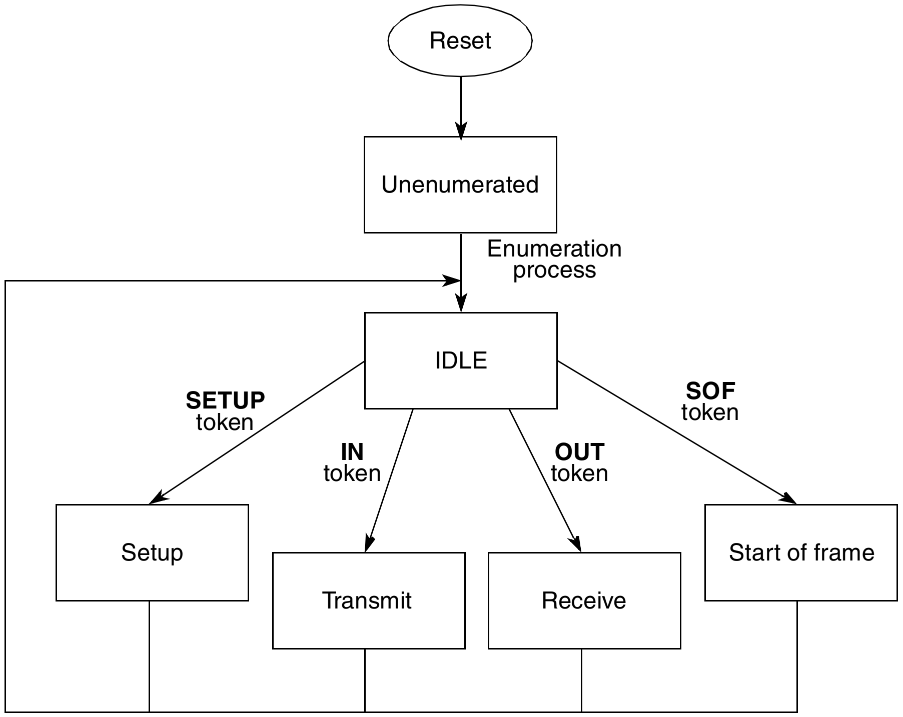


Figure 27-3. USB Controller Operating Modes

Table 27-2. 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 USEP<i>n</i>[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.
<b>USB Out Token Reception</b>
<table>
<tr><th>USEP<i>n</i>[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
(data discarded)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>11 (STALL)</td><td>No</td><td>STALL</td></tr>
</table>
</td></tr>
</table>

Table 27-2. USB Tokens (continued)

<table>
<tr><th>Token</th><th>Description</th></tr>
<tr><td>IN</td><td>To guarantee a transfer, the control software must preload the endpoint FIFO with a data packet before receiving an IN token. Software should set up the endpoint TxBD table and set USCOM[STR]. The USB controller fills the transmit FIFO and waits for the IN token. Once the token is received and the FIFO has been loaded with the last data byte or with at least 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 USEP<i>n</i>[THS] is set to respond with NAK, a NAK handshake is returned. If USEP<i>n</i>[THS] was set to respond with STALL, a STALL handshake is returned. (See table below.) When the end of the last buffer is reached (TxBD[L] is set), the CRC is appended. After the frame is sent, the USB controller waits for a handshake packet. If the host fails to acknowledge the packet, the timeout status bit TxBD[TO] is set. Software must set the proper DATA0/DATA1 PID in the transmitted packet.
<b>USB In Token Reception</b>
<table>
<tr><th>USEP<i>n</i>[THS]</th><th>FIFO Loaded</th><th>Handshake Sent to Host</th></tr>
<tr><td>00 (Normal)</td><td>No</td><td>NAK</td></tr>
<tr><td></td><td>Yes</td><td>Data packet is sent.</td></tr>
<tr><td>01 (Ignore)</td><td>—</td><td>None</td></tr>
<tr><td>10 (NAK)</td><td>—</td><td>NAK</td></tr>
<tr><td>11 (STALL)</td><td>—</td><td>STALL</td></tr>
</table>
</td></tr>
<tr><td>SETUP</td><td>The format of setup transactions is similar to OUT but uses a SETUP rather than an OUT PID. A SETUP token is recognized only by a control endpoint. When a SETUP token is received, setup reception begins. The USB controller fetches the next BD associated with the endpoint; if it is empty, the controller starts transferring the incoming packet to the buffer. When the buffer is full, the USB controller clears RxBD[E] and generates an interrupt if RxBD[I] = 1. If the incoming packet is larger than the buffer, the USB controller fetches the next BD and, if it is empty, continues transferring the rest of the packet to this buffer. The entire data packet including the DATA0 PID is written to the receive buffers. If the packet was received without CRC or bit stuff errors, an ACK handshake is sent to the host. If an error occurs, no handshake packet is returned and error status bits are set in the last RxBD associated with this packet.</td></tr>
<tr><td>Start of frame (SOF)</td><td>When an SOF packet is received, the USB controller issues a SOF maskable interrupt and the frame number entry in the parameter RAM is updated.</td></tr>
<tr><td>Preamble (PRE)</td><td>The PRE token signals the hub that a low-speed transaction is about to occur. The PRE token is read only by the hub. The USB controller ignores the PRE token function in function mode.</td></tr>
</table>

27.5 USB Host Description

When programmed as a host, the USB controller supports a limited host functionality. The following sections describe the available host functionality, its limitations, and the programming model.

Figure 27-4 illustrates the functionality of the USB controller in host mode. The USB controller consists of transmitter and receiver sections, a host control unit, and a function control unit, which is used for testing purposes. The USB transmitter contains four independent FIFOs, each containing 16 bytes. Endpoint 1 is dedicated for host transactions; endpoints 2-4 are for function transactions in test mode.

There is a dedicated FIFO for each of the four supported endpoints; endpoint 1 FIFO is for host transactions. The USB receiver has a single 16-byte FIFO.

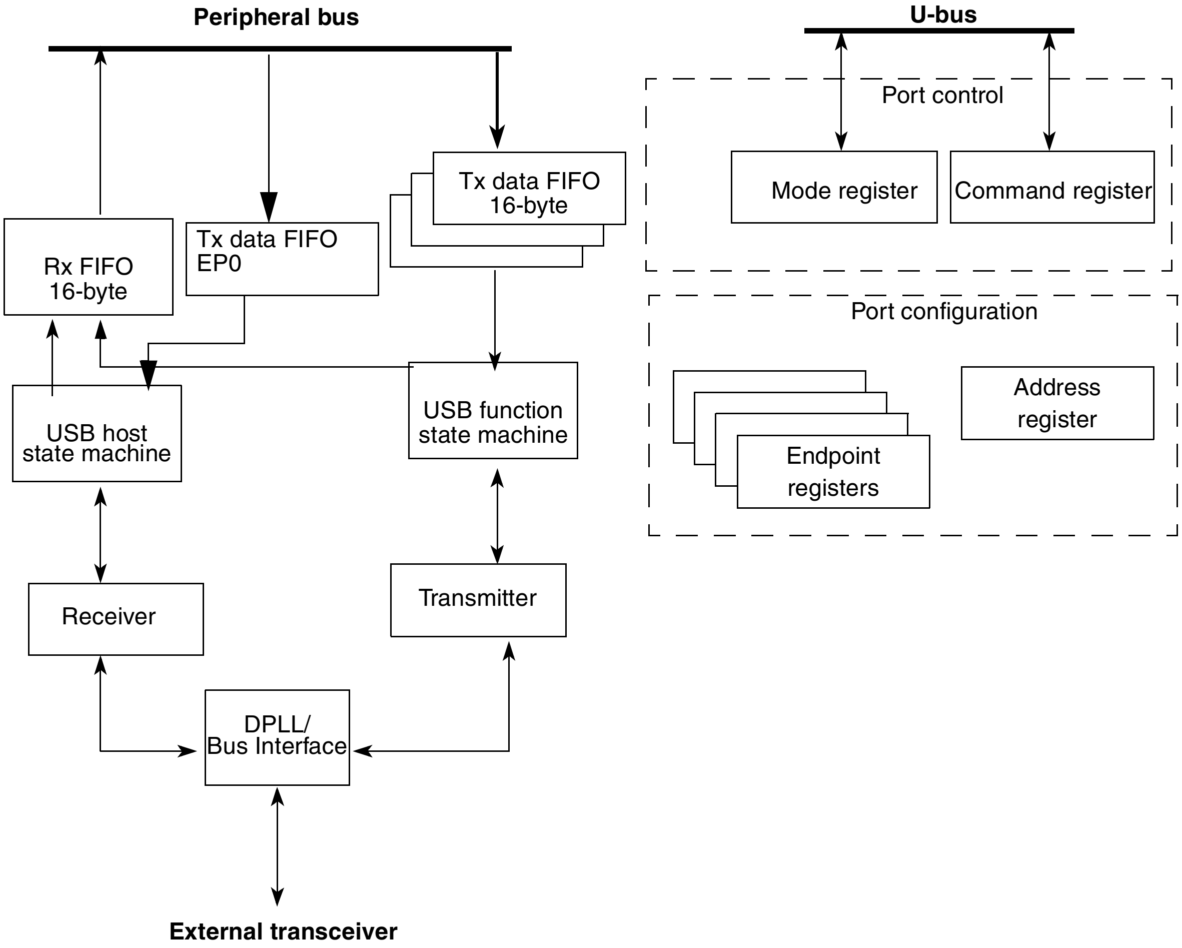


Figure 27-4. USB Controller Block Diagram

27.5.1 USB Host Controller Transmit/Receive

The USB host controller initiates all USB transactions in the system. After the reset condition, the HOST bit in USB mode register should be set (refer to Section 27.5.7.1, “USB Mode Register (USMOD)“) to enable host operation. USEP1 should be programmed for host operation as described in Section 27.5.7.3, “USB Endpoint Registers (USEP1–USEP4).”

After reset the host should enumerate the functions in the system. The enumeration process is done by software.

Once 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 starts. Figure 27-3 and Table 27-2 describe the behavior of the USB host controller for each transaction. Low speed transactions start with a preamble which is generated by the USB host controller state machine when the LSP bit in the TxBD is set.

When USMOD[TEST] is programmed, both the host state machine and function state machine are active. End points 2-4 receive/transmit data according to tokens received from host. The programming model and functional description are described in Section 27.5.7, "USB Function Programming Model."

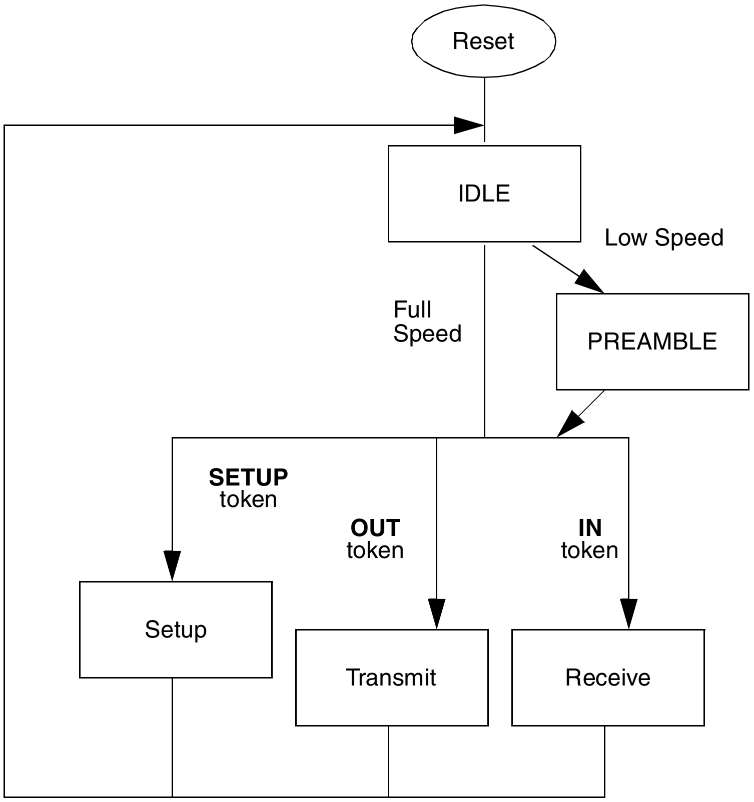


Figure 27-5. USB Controller Operating Modes

27.5.1.1 Packet-Level Interface

If USEP1[RTE] is 0, the USB host controller uses a packet-level interface to communicate with the user. Each transmit packet is prepared in a buffer and referenced by a TxBD as described in Section 27.6.3, "USB Transmit Buffer Descriptor (Tx BD) for Host." Each receive packet is stored in a buffer referenced by a RxBD as described in Section 27.6.1, "USB Receive Buffer Descriptor (Rx BD) for Host and Function." A SETUP or OUT transaction requires at least two TxBDs, one for the token and one or more for the data packet. An IN transaction requires one TxBD for the token and one or more RxBDs for the data packet. Tokens are not checked for validity and are transmitted as is. The user is responsible for token validity as well as CRC5 generation.

27.5.1.2 Transaction-Level Interface

NOTE

The transaction-level interface described in this section is available on .13µm (HiP7) Revision A.0 and future devices.

If USEP1[RTE] is 1, the USB host controller uses a transaction-level interface to communicate with the user. Each transaction uses one TrDB as described in Section 27.6.4, “USB Transaction Buffer Descriptor (TrBD) for Host.” The USB host controller generates the token based on the TOK field in the TrBD. For SETUP and OUT transactions, the TrBD points to a single buffer containing the data packet to be transmitted. For IN transactions, the TrBD points to a single buffer which is used for the receive data packet.

Table 27-3. USB Tokens

Token	Description	
OUT	Packet-Level Interface	Transaction-Level Interface
	Transmission begins when the USB host controller fetches a TxBD containing 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.	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.

USB Out Transaction

Token	Data	Handshake Received by Function	Indication on TxBD/TrBD
OUT	Sent by host	None	TO
		ACK	None
		NAK	NAK
		STALL	STALL

Table 27-3. USB Tokens (continued)

<table>
<tr><th>Token</th><th colspan="2">Description</th></tr>
<tr><td rowspan="2">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 correct DATA PID an RxBD is fetched. The received data and DATA PID are stored in receive FIFO. If RxBD[E] is set PID and data will be moved to the buffer. While receiving the data the USB host controller calculates CRC16, performs bit un-stuffing. On end of reception calculated CRC is compared to received and octet alignment is checked, RxBD[E] is cleared, RxBD[PID] is set according to received DATA PID and error indications are set if required:RxBD[CR] for failed CRC check, RxBD[NO] for non-octet sized data and RxBD[AB] if bit stuffing error occurred.
If no correct 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 correct 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="2"><b>USB In Transaction</b>
<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>

27.5.2 SOF Transmission for USB Host Controller

The following section describes the mechanism used by the USB Host Controller to support the automatic transmission of SOF tokens. This mechanism is enabled by setting USMOD[SFTE].

SOF packets should be transmitted every 1 ms. Since the time interval between two SOF packets must be more precise than could be accomplished by software, a hardware timer is used to assert an interrupt to the CP. When the interrupt is serviced by the CP, it prepares a SOF token and loads it to the host endpoint. Once the SOF token is loaded to the FIFO, it is transmitted like any other packet.

Before each SOF transmission, the software should prepare a value for the frame number and CRC5 to be transmitted in SOF token and place it in the parameter RAM (for further details please refer to Section 27.5.5, "Frame Number (FRAME_N)". One possible implementation would be to use the SOF interrupt (see Section 27.5.7.5, "USB Event Register (USBER)") to prepare the frame number for the next SOF packet. The SFT interrupt should not be used for this purpose since it is generated before the SOF packet is actually transmitted.

The application software should also guarantee that the USB host has completed all pending transactions prior to the 1 ms tick, so that the transmit FIFO is empty at this point. The current value of the SOF timer may be read at any time to synchronize the software with the USB frames. See Section 27.5.7.8, "USB Start of Frame Timer (USSFT)" for more information.

27.5.3 USB Function and Host Parameter RAM Memory Map

The USB controller parameter RAM area, shown in Table 27-4, 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 27-4. USB Parameter RAM Memory Map

Address	Name[1]	Width	Description
USB Base + 00	**EP1PTR**	Half Word	Endpoint pointer registers 1–4. 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 27-6. The map of the endpoint parameter block is shown in Table 27-5 **Note**: When USB host mode is set, **EP1PTR** must be used for the host end point.
USB Base + 02	**EP2PTR**	Half Word	
USB Base + 04	**EP3PTR**	Half Word	
USB Base + 06	**EP4PTR**	Half Word	
USB Base + 08	RSTATE	Word	Receive internal state. Reserved for CP use only. Should be cleared before enabling the USB controller.
USB Base + 0C	RPTR	Word	Receive internal data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
USB Base + 10	**FRAME_N**	Half Word	Frame number. See Figure 27-7 **Note**: The definition of this parameter is different for host mode and function mode.
USB Base + 12	RBCNT	Half Word	Receive internal byte count. A down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.
USB Base + 14	RTEMP	Word	Receive temp. Reserved for CP use only.

Table 27-4. USB Parameter RAM Memory Map

Address	Name[1]	Width	Description
USB Base + 18	RXUSB_ Data	Word	Rx Data temp
USB Base + 1C	RXUPTR	Half Word	Rx microcode return address temp

[1] The items in **boldface** should be initialized by the user before the USB controller is enabled; other values are initialized by the CP.

Once initialized, the parameter RAM values do not normally need to be accessed by user software. They should only be modified when no USB activity is in progress.

27.5.4 Endpoint Parameters Block Pointer (EPxPTR)

The endpoint parameter block pointers (EPxPTR) are DPRAM indices to an endpoint's parameter block. The parameter block can be allocated to any address that is divisible by 32. The format of the endpoint pointer registers (EP*x*PTR) is shown in Figure 27-6.

	0 … 10	11 … 15
Field	Endpoint Index Pointer	—
R/W	R/W	
Reset	—	
Addr	USB base + 0x00 (EP1PTR), 0x02 (EP2PTR), 0x04 (EP3PTR), 0x06 (EP4PTR)	

Figure 27-6. Endpoint Pointer Registers (EP*x*PTR)

The map of the endpoint parameter block is shown in Table 27-5.

Table 27-5. 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 using the transaction-level interface in host mode, TBASE points to the TrBD ring, and RBASE is unused.
0x02	**TBASE**	16 bits	
0x04	**RFCR**	8 bits	Rx/Tx function code. Controls the value to appear on AT[1–3] when the associated SDMA channel accesses memory and the byte-ordering convention.
0x05	**TFCR**	8 bits	

Table 27-5. Endpoint Parameter Block (continued)

Offset[1]	Name[2]	Width	Description
0x06	**MRBLR**	16 bits	Maximum receive buffer length. Defines the maximum number of bytes the MPC8272 writes to the USB receive buffer before moving to the next buffer. MRBLR must be divisible by 4. The MPC8272 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 CP and does not have to be initialized by the user.
0x08	**RBPTR**	16 bits	RxBD pointer. Points to the next BD the receiver will transfer data to when it is in an idle state or to the current BD while processing a frame. Software should initialize RBPTR after reset. When the end of the BD table is reached, the CP initializes this pointer to the value programmed in RBASE. Although the user does not need to write RBPTR in most applications (except initialization), it can be changed when the receiver is disabled or when no receive buffer is being used. When using the transaction-level interface in host mode, this field is unused.
0X0A	**TBPTR**	16 bits	TxBD pointer. Points to the next BD that the transmitter will transfer data from when it is in an idle state or to the current BD during frame transmission. TBPTR should be initialized by the software after reset. When the end of BD table is reached, the CP initializes this pointer to the value programmed in the TBASEn entry. Although the user never needs to write TBPTR, in most applications (except initialization), it can be changed when the transmitter is disabled or when no transmit buffer is being used.
0X0C	TSTATE[3]	32 bits	Transmit internal state. Reserved for CP use only. Should be cleared before enabling the USB controller.
0x10	TPTR[3]	32 bits	Transmit internal data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x14	TCRC[3]	16 bits	Transmit temp CRC. Reserved for CP use only.
0x16	TBCNT[3]	16 bits	Transmit internal byte count. A down-count value that is initialized with the TxBD data length and decremented with every byte read by the SDMA channels.
0x18	TTEMP	32 bits	Tx temp
0x1C	TXUSBU_PTR	16 bits	Tx microcode return address temp
0x1E	**HIMMR**	16 bits	When using the transaction-based interface in host mode, this field must be programmed to match the high 16 bits of the IMMR. Otherwise, this field is unused.

[1] Offset from endpoint parameter block base.

[2] Note that the items in **boldface** should be initialized by the user.

[3] These parameters need not be accessed in normal operation but may be helpful for debugging.

27.5.5 Frame Number (FRAME_N)

This entry is used for frame number updates both in function mode and in host mode. In function mode it is written by the USB controller; in host mode it is written by the application software and USB controller.

This entry is updated by the USB controller whenever a 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.

	0	1	4	5	15
Field	V[1]	—		FRAME NUMBER	
Reset	—				
R/W	R/W				
Addr	USB base + 0x10				

Figure 27-7. Frame Number (FRAME_N) in Function Mode— Updated by USB Controller

[1] This bit is set if the SOF token was received error free.

Table 27-6 describes FRAME_N fields.

Table 27-6. 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 27.5.1.2, "Transaction-Level Interface" for details. The software should prepare the frame number and the CRC and place it in FRAME_N field.

	0	1	4	5	15
Field	CRC5			FRAME NUMBER	
Reset	—				
R/W	R/W				
Addr	USB base + 0x10				

Figure 27-8. Frame Number (FRAME_N) in Host Mode— Updated by Application Software

Table 27-6 describes FRAME_N fields.

Table 27-7. 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.

27.5.6 USB Function Code Registers (RFCR and TFCR)

RFCR and TFCR control the value that the user would like to appear on the Address Type pins (AT1–AT3) when the associated SDMA channel accesses memory.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	DTB	—

Figure 27-9. USB Function Code Registers (RFCR and TFCR)

Table 27-8 describes RFCR and TFCR fields.

Table 27-8. RFCR and TFCR Fields

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global 0 Snooping disabled 1 Snooping enabled
3–4	BO	Byte ordering. This bit field should be set by the user to select the required byte ordering for the data buffer. If this bit field is modified on the fly, it will take effect at the beginning of the next frame. 00 DEC (and Intel) convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The transmission order of bytes within a buffer word is reversed as compared to the Freescale mode. This mode is supported only for 32-bit port size memory. 01 PowerPC little-endian byte ordering. As data is transmitted onto the serial line from the data buffer, the least significant byte of the buffer double word contains data to be transmitted earlier than the most-significant byte of the same buffer double word. 1X Freescale byte ordering—normal operation. It is also called 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	TC2	Transfer code. Contains the transfer code value of TC[2] used during this SDMA channel memory access.TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access
6	DTB	Data bus Indicator 0 Use 60x bus for SDMA operation 1 Reserved
7	—	Reserved, should be cleared.

27.5.7 USB Function Programming Model

The following sections describe USB controller registers.

27.5.7.1 USB Mode Register (USMOD)

USMOD, shown in Figure 27-10, 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	0x11B60							

Figure 27-10. USB Mode Register (USMOD)

Table 27-9 describes USMOD fields.

Table 27-9. USMOD Fields

Bits	Name	Description
0	LSS	Low-speed signaling. Selects the signaling speed. The actual bit rate depends on the USB clock source. 0 Full-speed (12-Mbps) signaling. Normal operation. 1 Low-speed (1.5-Mbps) signaling. For a point-to-point connection with a low-speed device or for local loopback testing.
1	RESUME	Generate resume condition. When set, this bit generates a resume condition on the USB. This bit should be used if the function wants to exit the suspend state.
2–3	—	Reserved, should be cleared.
4	SFTE	Start-of-frame timer enable. Setting this bit enables the Start-of-Frame timer and automatic SOF transmission. See Section 27.5.7.8, “USB Start of Frame Timer (USSFT)”, and Section 27.5.2, “SOF Transmission for USB Host Controller”, for more information. 0 SOF timer is disabled 1 SOF timer is enabled **Note:** When SFTE is 1, the PC21 pin cannot be used as CP_INT since the CP interrupt is used internally for generating the SOF packet.
5	TEST	USB controller test (loopback) mode 0 Test mode is disabled 1 Test mode is enabled **Note:** This bit may be set only when HOST is set (USB host mode)
6	HOST	USB host mode 0 USB host disabled 1 USB host 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.

27.5.7.2 USB Slave Address Register (USADR)

The USB address register is an 8-bit, memory-mapped register. It holds the address for this USB port when operating as function.

	0	1 7
Field	—	SADx
Reset	0000_0000	
R/W	R/W	
Addr	0x11B61	

Figure 27-11. USB Slave Address Register (USADR)

Table 27-10 describes USADR fields.

Table 27-10. 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

27.5.7.3 USB Endpoint Registers (USEP1–USEP4)

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	0x11B64 (USEP1); 0x11B66 (USEP2); 0x11B68 (USEP3); 0x11B6A (USEP4)							

Figure 27-12. USB Endpoint Registers (USEP1–USEP4)

Table 27-11 describes the fields of USEP1–USEP4. The setting for USB host controller should be set only in USEP1, when USMOD[HOST] is set.

Table 27-11. USEPx Field Descriptions

Bits	Name	USB Function Mode	USB Host Mode
0–3	EPN	Endpoint number. For USB **function** controller defines the supported endpoint number.	For USB **host** controller, should be cleared
4–5	—	Reserved, should be cleared	Reserved, should be cleared
6–7	TM	Transfer mode for USB **function** controller 00 Control 01 Interrupt 10 Bulk 11 Isochronous	Transfer mode for USB **host** controller 00 Control /interrupt/bulk 11 Isochronous
8–9	—	Reserved, should be cleared.	Reserved, should be cleared

Table 27-11. USEPx Field Descriptions (continued)

Bits	Name	USB Function Mode	USB Host Mode
10	MF	Enable multi-frame. For USB **function** controller allows loading of the next transmit packet into the FIFO before transmission completion of the previous packet. 0 Transmit FIFO may hold only one packet 1 Transmit FIFO may hold more than one packet **Note:** For USB function configuration: Should be cleared unless the endpoint is configured for ISO transfer mode.	Enable multi-frame for USB **host** controller. Should be always set.
11	RTE	Retransmit enable for USB **function** controller 0 No retransmission 1 Automatic frame retransmission is enabled. The frame will be retransmitted if transmit error occurred (time-out). **Note:** May be set only if the transmit packet is contained in a single buffer. If it is not, retransmission should be handled by software intervention. **Note:** Should be set to zero for endpoint that is configured for ISO transfer mode	Transaction-level interface for USB **host** controller. 0 Packet-level interface as described in Section 27.5.1.1, “Packet-Level Interface.” 1 Transaction-level interface as described in Section 27.5.1.2, “Transaction-Level Interface.”
12–13	THS	Transmit hand shake 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; in this case THS will be cleared by the USB function controller when a SETUP token is received.	Transmit hand shake for USB **host** controller 00 Normal handshake
14–15	RHS	Receive hand shake 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 may be used for flow control. Not allowed for control endpoint. 11 Force STALL handshake. On a control endpoint this value is used to generate a protocol stall; in this case RHS will be cleared by the USB function controller when a SETUP token is received.	Receive hand shake for USB **host** controller 00 Normal handshake

27.5.7.4 USB Command Register (USCOM)

USCOM is used to start USB transmit operation.

	0	1	2	3	4	5	6	7
Field	STR	FLUSH	ISFT	DSFT	—		EP	
Reset	0000_0000							
R/W	R/W							
Addr	0x11B62							

Figure 27-13. USB Command Register (USCOM)

Table 27-12 describes USCOM fields.

Table 27-12. USCOM Fields

Bits	Name	Description
0	STR	Start FIFO fill. Setting the STR bit to one causes the USB controller to start the filling the corresponding end point transmit FIFO with data. Transmission will begin once the IN token for this end-point is received. The STR bit is read always as a zero.
1	FLUSH	Flush FIFO. Setting the FLUSH bit to one causes the USB controller to flush the corresponding end point transmit FIFO. Before flushing the FIFO, the user should issue the Stop_Tx command. After flushing the FIFO the user should issue the Restart_Tx command (Refer to Section 27.7, “USB CP Commands.”). FLUSH is always read as a zero.
2	ISFT	Increment Start-of-Frame Time. Setting the ISFT bit increments the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
3	DSFT	Decrement Start-of-Frame Time. Setting the DSFT bit decrements the start-of-frame time by one. This bit could be used to synchronize the USB frames to an external timing source.
4–5	—	Reserved, should be cleared.
6–7	EP	End point. Selects one of the four supported end points.

27.5.7.5 USB Event Register (USBER)

The USBER reports events recognized by the USB channel and generates interrupts. Upon recognition of an event, the USB sets its corresponding bit in the USBER. Interrupts generated by this register may be masked in the USB mask register.

The USBER may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit’s value). More than one bit may be cleared at a time. All unmasked bits must be cleared before the CP will clear the internal interrupt request. This register is cleared at reset.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—					SFT	RESET	IDLE	TXE4	TXE3	TXE2	TXE1	SOF	BSY	TXB	RXB
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11B70															

Figure 27-14. USB Event Register (USBER)

Table 27-13 describes USBER fields.

Table 27-13. USBER Fields

Bit	Name	Description
0 –4	—	Reserved, should be cleared.
5	SFT	The start-of-frame timer (USSFT[SFT]) wrapped from 11,999 to 0.
6	RESET	Reset condition detected. USB reset condition was detected asserted.
7	IDLE	IDLE status changed. A change in the status of the serial line was detected. The real time suspend status is reflected in the USB status register.
8–11	TXEx	Tx error. An error occurred during transmission for End Point x (packet not acknowledged or underrun).
12	SOF	Start of frame. A start of frame packet was received. The packet is stored in the FRAME_N parameter ram entry.
13	BSY	Busy condition. Received data has been discarded due to a lack of buffers. This bit is set after the first character is received for which there is no receive buffer available.
14	TXB	Tx buffer. A buffer has been transmitted. This bit is set once the transmit data of the last character in the buffer was written to the transmit FIFO (if L=0 (last bit)) or after the last character was transmitted on the line (if L=1).
15	RXB	Rx buffer. A buffer has been received. This bit is set after the last character has been written to the receive buffer and the Rx BD is closed.

27.5.7.6 USB Mask Register (USBMR)

The USBMR is a 16-bit read/write register (0x11B74) that has the same bit formats as the USB event register. If a bit in the USBMR is one, the corresponding interrupt in the USBER is enabled. If the bit is zero, the corresponding interrupt in the USBER will be masked. This register is cleared at reset.

27.5.7.7 USB Status Register (USBS)

The USB status register, described in Figure 27-15 and Table 27-14, is a read-only register that allows the user to monitor real-time status condition on the USB lines.

	0	1	2	3	4	5	6	7
Field	—							IDLE
Reset	0000_0000							
R/W	R							
Addr	0x11B77							

Figure 27-15. USB Status Register (USBS)

Table 27-14 describes USBS fields.

Table 27-14. USBS Fields

Bit	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.

27.5.7.8 USB Start of Frame Timer (USSFT)

NOTE

> The USSFT described in this section is available only on .13 µm (HiP7) Revision A.0 and future devices.

When enabled by USMOD[SFTE], the USSFT contains the current time within the frame with a resolution of one bit time. When the value of USSFT wraps from 11,999 to 0, a CP interrupt is asserted to trigger the transmission of a SOF packet, and USBER[SFT] is set

The USSFT may be read at any time.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field			SFT													
Reset	0010_1110_1101_1000															
R/W	R															
Addr	0x11B78															

Figure 27-16. USB Start of Frame Timer (USSFT)

Table 27-13 describes USSFT fields.

Table 27-15. USSFT Fields

Bit	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.

27.6 USB Buffer Descriptor Ring

The data associated with the USB channel is stored in buffers that are referenced by BDs organized in BD rings located in the dual-port RAM (refer to Figure 27-17). These rings have the same basic configuration as those used by the SCCs and SMCs.

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 CP confirms reception and transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced.

The buffers may reside in either external or internal memory.

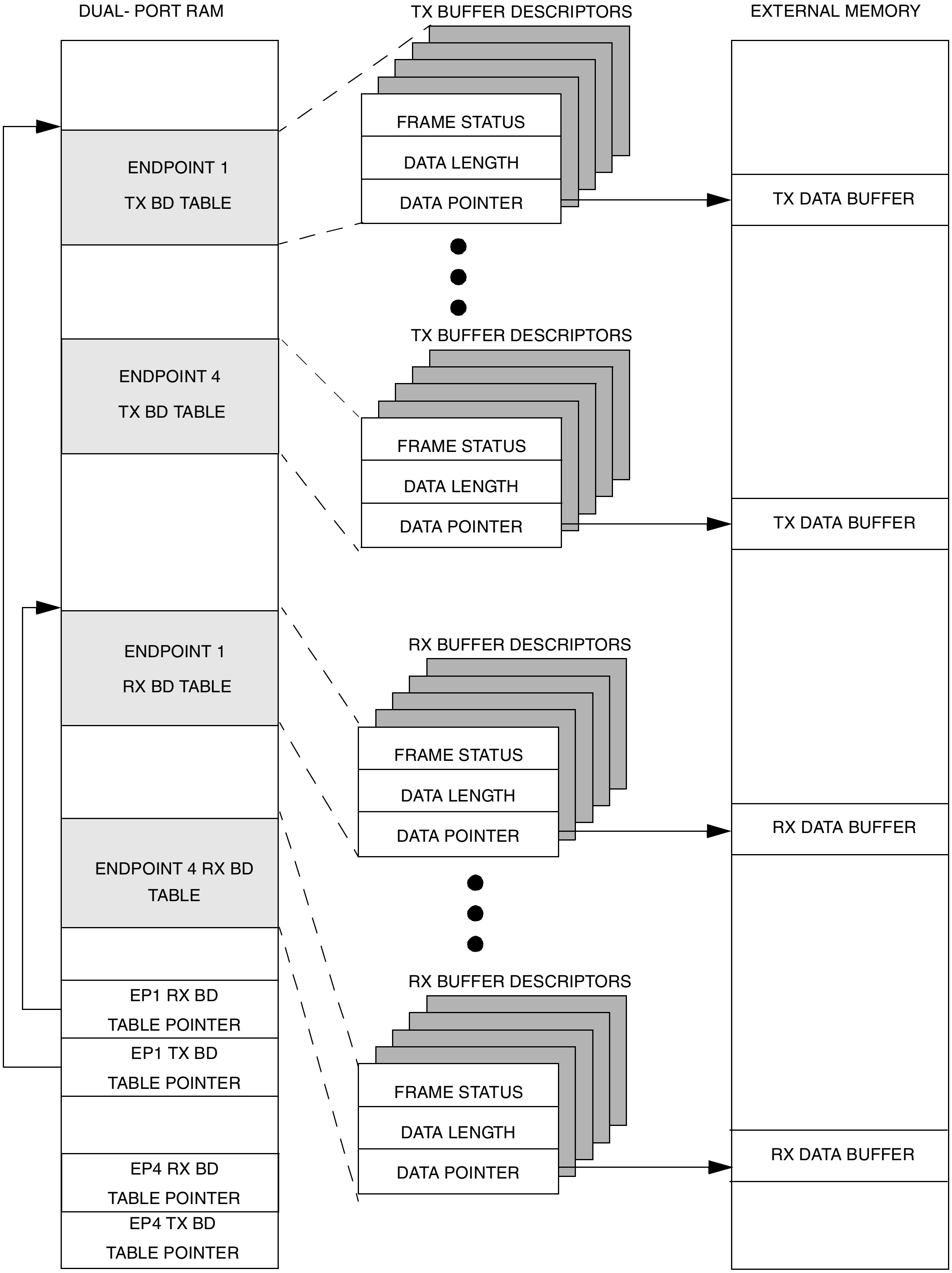


Figure 27-17. USB Memory Structure

27.6.1 USB Receive Buffer Descriptor (Rx BD) for Host and Function

The CP reports information about each buffer of received data using Rx BDs. The CP closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer when the current buffer is full. Additionally, it closes the buffer on the following conditions:

- End of packet detected
- Overrun error occurred
- Bit stuff violation detected

As shown in Figure 27-18, the first word of the Rx BD contains status and control bits. These bits are prepared by the user before reception and are set by the CP after the buffer has been closed. The second word contains the data length—in bytes—that was received. The third and fourth words contain a pointer that always points to the beginning of the received data buffer.

The RxBD is identical for both the host mode (when using the packet-level interface) and the function mode. There are no RxBDs in host mode when using the transaction-level interface.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0	**E**	—	**W**	**I**	L	F	—		PID		—	NO	AB	CR	OV	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

Figure 27-18. USB Receive Buffer Descriptor (Rx BD)[1,2]

[1] Entries in **boldface** must be initialized by the user.

[2] All fields should be written by the CPU core before enabling the USB.

Table 27-16 describes USB receive buffer descriptor fields.

Table 27-16. USB Rx BD Fields

Offset	Bit	Name	Description
0x00	0	**E**	Empty 0 The data buffer associated with this Rx BD 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 Rx BD. The CP will not use this BD again while the E-bit remains zero. 1 The data buffer associated with this BD is empty, or reception is currently in progress. This Rx BD and its associated receive buffer are owned by the CP. Once the E-bit is set, the CPU core should not write any fields of this Rx BD.
	1	—	Reserved, should be cleared
	2	**W**	Wrap (Final BD in table) 0 This is not the last BD in the Rx BD table. 1 This is the last BD in the Rx BD table. After this buffer has been used, the CP will receive incoming data into the first BD in the table (the BD pointed to by RBASE). The number of Rx BDs in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

Table 27-16. USB Rx BD Fields (continued)

Offset	Bit	Name	Description
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been filled. 1 The RXB bit in the USB event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU core to process the buffer. The RXB bit can cause an interrupt if it is enabled.
	4	L	Last. This bit is set by the USB controller when the buffer is closed due to detection of end-of-packet condition on the bus, or as a result of error. Written by the USB controller after the received data has been placed into the associated data buffer. 0 Buffer does not contain the last byte of the message. 1 Buffer contains the last byte of the message.
	5	F	First. This bit is set by the USB controller when the buffer contains the first byte of a packet. Written by the USB controller after the received data has been placed into the associated data buffer. 0 Buffer does not contain the first byte of the message 1 Buffer contains the first byte of the message
	6–7	—	Reserved, should be cleared
	8–9	PID	Packet ID. This bit field is set by the USB controller to indicate the type of the packet. This bit is valid only if the USB RXBD[F] is set. Written by the USB controller after the received data has been placed into the associated data buffer. 00 Buffer contains DATA0 packet 01 Buffer contains DATA1 packet 10 Buffer contains SETUP packet. This option can never be set on host RxBD
	10	—	Reserved, should be cleared
	11	NO	Rx non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	AB	Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	13	CR	CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer. Written by the USB controller after the received data has been placed into the associated data buffer.
	14	OV	Overrun. A receiver overrun occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	15	—	Reserved, should be cleared
0x02	0–15	Data length	Data length is the number of octets that the CP has written into this BD's data buffer. It is written once by the CP as the BD is closed. **Note:** The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.
0x04	0–31	**Rx data buffer pointer**	The receive buffer pointer, which always points to the first location of the associated data buffer, must be divisible by 4. The buffer may reside in either internal or external memory

Data length represents the number of octets that the CP has written into this BD's buffer. It is written once by the CP as the BD is closed.

The receive buffer pointer always points to the first location of the associated buffer. The pointer must be divisible by 4. The buffer may reside in either internal or external memory.

27.6.2 USB Transmit Buffer Descriptor (Tx BD) for Function

Data that the USB function wishes to transmit to the host is arranged in buffers referenced by the Tx BD ring. The first word of the Tx BD contains the 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**		**PID**		—			TO	UN	—
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

Figure 27-19. USB Transmit Buffer Descriptor (Tx BD)[1,2]

[1] Entries in **boldface** must be initialized by the user.

[2] All fields should be prepared by the user before transmission.

Table 27-17 describes USB TxBD fields.

Table 27-17. USB Function Tx BD Fields

Offset	Bit	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared
	2	**W**	Wrap (Final BD in table) 0 This is not the last BD in the Tx BD table. 1 This is the last BD in the Tx BD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASEx). The number of Tx BDs in this table is programmable, and is determined only by the Tx BD[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 27-17. USB Function Tx BD Fields (continued)

Offset	Bit	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 27.5.7.3, “USB Endpoint Registers (USEP1–USEP4).” 0 Continue to load the transmit FIFO with the next packet. Several packets may be loaded to the FIFO. 1 Last packet that is loaded to FIFO. No more packets will be loaded to fifo after a packet marked CNF, till it transmitted.
	7		Reserved, should be cleared
	8–9	**PID**	Packet ID. This bit field is valid for the first BD of a packet; otherwise it is ignored. 0X Do not append PID to the data. 10 Transmit DATA0 PID before sending the data. 11 Transmit DATA1 PID before sending the data.
	10–12	—	Reserved, should be cleared
	13	TO	Time out. Indicates that the host failed to acknowledge the packet.
	14	UN	Underrun. Indicates that the USB encountered a transmitter underrun condition while sending the buffer.
	15	—	Reserved, should be cleared
0x02	0–15	**Data length**	The data length is the number of octets that the CP should transmit from this BD’s data buffer. It is never modified by the CP. This value should normally be greater than zero.
0x04	0–31	**Tx data buffer pointer**	The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd.The buffer may reside in either internal or external memory.

Data length (the second half word of a TxBD) is the number of octets the CP should send from this BD’s data buffer. It is never modified by the CP.

Tx buffer pointer (the third and fourth half words of a TxBD) always points to the first location of the buffer in internal or external memory. The pointer may be even or odd.

27.6.3 USB Transmit Buffer Descriptor (Tx BD) for Host

The Tx BD described in this section is used when the packet-level interface is active. See Section 27.5.1.1, “Packet-Level Interface” for more information.

Data to be transmitted with the USB to the CP by is arranged in buffers referenced by the Tx BD ring. The first word of the Tx BD 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**															

Figure 27-20. USB Transmit Buffer Descriptor (Tx BD)[1,2]

OFFSET + 4	**TX DATA BUFFER POINTER**
OFFSET + 6	

Figure 27-20. USB Transmit Buffer Descriptor (Tx BD)[1,2]

[1] Entries in **boldface** must be initialized by the user.

[2] All fields should be prepared by the user before transmission.

Table 27-17 describes USB TxBD fields.

Table 27-18. USB Host Tx BD Fields

Offset	Bit	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared
	2	**W**	Wrap (Final BD in Table) 0 This is not the last BD in the Tx BD table. 1 This is the last BD in the Tx BD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASEx). The number of Tx BDs in this table is programmable, and is determined only by the Tx BD[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. 1The 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.
	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 27.5.7.3, “USB Endpoint Registers (USEP1–USEP4).” 0 Continue to load the transmit FIFO with the next packet. No handshake or response is expected from the function for this packet. 1 Wait for handshake or response from the function before starting the next packet, or this is the last packet. Do not clear CNF for a token preceding a data packet unless the data packet’s BD is ready.
	7	LSP	Low-speed transaction. Use for tokens only. 0 The following transaction is with the host or a full-speed device. 1 The following transaction is with a low-speed device. Required only for tokens. Note that LSP should always be cleared in slave mode.

Table 27-18. USB Host Tx BD Fields (continued)

Offset	Bit	Name	Description
	8–9	**PID**	Packet ID. This bit field is valid for the first BD of a packet; otherwise it is ignored. 0X Do not append PID to the data. 10 Transmit DATA0 PID before sending the data. 11 Transmit DATA1 PID before sending the data.
	10	—	Reserved, should be cleared
	11	NAK[1]	NAK received. Indicates that the endpoint has responded with a NAK handshake. The packet was received error-free; however, the endpoint could not accept it.
	12	STAL[1]	STALL received. Indicates that the endpoint has responded with a STALL handshake. The endpoint needs attention through the control pipe.
	13	TO[1]	Time out. Indicates that the endpoint failed to acknowledge the packet.
	14	UN[1]	Underrun. Indicates that the USB encountered a transmitter underrun condition while sending the buffer.
	15	—	Reserved, should be cleared
0x02	0–15	**Data length**	The data length is the number of octets that the CP should transmit from this BD's data buffer. It is never modified by the CP. This value should normally be greater than zero.
0x04	0–31	**Tx data buffer pointer**	The transmit buffer pointer, which always points to the first location of the associated data buffer, may be even or odd.The buffer may reside in either internal or external memory.

[1] Written by the USB controller after it finishes sending the associated data buffer.

Data length (the second half word of a TxBD) is the number of octets the CP should send from this BD's data buffer. It is never modified by the CP.

Tx buffer pointer (the third and fourth half words of a TxBD) always points to the first location of the buffer in internal or external memory. The pointer may be even or odd.

27.6.4 USB Transaction Buffer Descriptor (TrBD) for Host

The TrBD described in this section is used when the transaction-level interface is active. See Section 27.5.1.2, "Transaction-Level Interface" for more information.

Data to be transmitted with the USB to the CP by is arranged in buffers referenced by the TrBD ring. The first word of the TrBD contains status and control bits.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OFFSET + 0x0	**R**	—	**W**	**I**	**L**	**TC**	**CNF**	**LSP**	**PID**		RXER	NAK	STAL	TO	UN	BOV
OFFSET + 0x2	**DATA LENGTH**															
OFFSET + 0x4	**DATA BUFFER POINTER**															
OFFSET + 0x6																
OFFSET + 0x8	**TOK**		—	**ISO**	—	**ENDP**				**ADDR**						

Figure 27-21. USB Transaction Buffer Descriptor (TrBD)[1,2]

OFFSET + 0xA	Reserved

Figure 27-21. USB Transaction Buffer Descriptor (TrBD)[1,2]

[1] Entries in **boldface** must be initialized by the user.

[2] All fields should be prepared by the user before transmission.

Table 27-17 describes USB TrBD fields.

Table 27-19. USB Host TrBD Fields

Offset	Bit	Name	Description
0x00	0	**R**	Ready 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TrBD table. 1 This is the last BD in the TrBD table. After this buffer has been used, the CP will send data using the first BD in the table (the BD pointed to by TBASE). The number of TrBDs in this table is programmable, and is determined only by the TrBD[W] and the overall space constraints of the dual-port RAM.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 The TXB bit in the event register is set when this buffer is serviced. TXB can cause an interrupt if it is enabled.
	4	**L**	Last This bit should always be 1 since each TrBD represents an entire transaction.
	5	**TC**	Transmit CRC. Append CRC to transmitted data packet. 0 Transmit end-of-packet after the last data byte. This setting can be used for testing purposes to send a bad CRC after the data. 1 Transmit the CRC sequence after the last data byte.
	6	**CNF**	Transmit confirmation. This bit should always be set to 1 to obtain confirmation for each transaction.
	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 27-19. USB Host TrBD Fields (continued)

Offset	Bit	Name	Description
	8–9	**PID**	Packet ID. For OUT/SETUP transactions, this field is prepared by the user with the following values: 0X Do not append PID to the data packet. 10 Transmit DATA0 PID before sending the data packet. 11 Transmit DATA1 PID before sending the data packet. For IN transactions, this field is provided by the USB host controller with the following values: 00 Buffer contains DATA0 packet. 01 Buffer contains DATA1 packet.
	10	RXER[1]	Receive Error. This bit indicates that an error was detected while receiving the data packet of an IN transaction. If RXER is 1, bits 11-15 have a different meaning as explained below.
	11	NAK/NO[1]	RXER=0: NAK received. Indicates that the endpoint has responded with a NAK handshake (OUT transaction). The packet was received error-free; however, the endpoint could not accept it. RXER=1: Rx non-octet aligned packet. A packet that contained a number of bits not exactly divisible by eight was received. Written by the USB controller after the received data has been placed into the associated data buffer.
	12	STAL/AB[1]	RXER=0: STALL received. Indicates that the endpoint has responded with a STALL handshake (OUT transaction). The endpoint needs attention through the control pipe. RXER=1: Frame aborted. Bit stuff error occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	13	TO/CR[1]	RXER=0: Time out. Indicates that the endpoint failed to acknowledge the token (IN transaction) or the data packet (OUT/SETUP transaction). RXER=1: CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer. Written by the USB controller after the received data has been placed into the associated data buffer.
	14	UN/OV[1]	RXER=0: Underrun. Indicates that the USB encountered a transmit FIFO underrun condition while sending the data packet (OUT/SETUP transaction). RXER=1: Overrun. An internal receive FIFO overrun occurred during reception. Written by the USB controller after the received data has been placed into the associated data buffer.
	15	BOV[1]	Buffer Overflow. IN transactions only. Indicates that the number of received bytes is larger than the buffer size as provided in the Data Length field.
0x02	0–15	**Data Length**	For OUT/SETUP transactions, the user prepares this field with the number of bytes to be sent from the data buffer. It will not be modified by the CP. For IN transactions, the user prepares this field with the size of the data buffer, which must be divisible by 4. The CP will return the actual number of bytes written to the data buffer. If the number of received bytes, including the 2-byte CRC, is larger than the data buffer, the BOV bit will be set by the CP.

Table 27-19. USB Host TrBD Fields (continued)

Offset	Bit	Name	Description
0x04	0–31	**Data Buffer Pointer**	The data buffer pointer. The buffer may reside in either internal or external memory. For OUT/SETUP transactions, this points to the buffer containing the data packet to transmit. It may have any alignment. For IN transactions, this points to the buffer into which the data packet should be received, The pointer must be divisible by 4.
0x08	0-1	**TOK**	Token Type This field determines the type of token to be transmitted and the type of transaction. 00 SETUP 01 OUT 10 IN 11 Reserved
	2	—	Reserved, should be cleared.
	3	**ISO**	Isochronous This bit indicates that the transaction is isochronous, so no handshake is required. 0 Bulk/Control/Interrupt. The handshake packet is automatically expected or generated by the USB Host Controller. 1 Isochronous. No handshake packets are expected or generated. This bit actually controls the value that is written to USEP1[TM] before processing this transaction.
	5	—	Reserved, should be cleared.
	5-8	**ENDP**	Endpoint This field indicates the endpoint number to be included in the token.
	9-15	**ADDR**	Address This field indicates the device address to be included in the token.
0x0A	0-15	—	Reserved, should be cleared.

[1] Written by the USB controller after it finishes sending or receiving the associated data buffer.

27.7 USB CP Commands

The following transmit commands are issued to the CP command register (CPCR). Refer to Section 13.4.1, “CP Command Register (CPCR).”

27.7.1 STOP Tx Command

This command disables the transmission of data on the selected endpoint. After issuing the command the corresponding End Point FIFO should be flushed. No further transmissions will take place until the Restart Tx Command is issued.

27.7.2 RESTART Tx Command

This command enables the transmission of data from the corresponding endpoint on the USB. This command is expected by the USB controller after a STOP Tx Command, or after transmission error (underrun or time-out).

27.8 USB Controller Errors

The USB controller reports frame reception and transmission error conditions using the BDs and the USB event register (USBER). Transmission errors are shown in Table 27-20. Errors which exist exclusively in host mode or function mode are marked as such.

Table 27-20. USB Controller Transmission Errors

Error	Description
Transmit Underrun	If an underrun occurs, the transmitter forces a bit stuffing violation, terminates buffer transmission, closes the buffer, sets TxBD[UN] and the corresponding USBER[TXE*n*]. The endpoint resumes transmission after the RESTART TX ENDPOINT command is received.
Transmit Timeout	Transmit packet not acknowledged. If a timeout occurs, the controller tries to retransmit if USEP*n*[RTE] = 1. If RTE = 0 or the second attempt fails, the controller closes the buffer and sets TxBD[TO] and USBER[TXE*n*]. The endpoint resumes transmission after receiving a RESTART TX ENDPOINT command.
Tx Data Not Ready	For **USB function mode** only. This error occurs if an IN token is received, but the corresponding endpoint's transmit FIFO is empty, or if the target endpoint is configured to NAK or STALL. The controller sets USBER[TXE*n*].
Reception of NAK or STALL hand shake	For **USB host mode** only. If this error occurs, the channel closes the buffer, sets the corresponding status bit in the Tx BD (NAK or STAL), and sets the TXE bit in the USB event register. When using the packet-level interface, the host will resume transmission after reception of the RESTART TRANSMIT command.

Table 27-21 describes the USB controller reception errors.

Table 27-21. USB Controller Reception Errors

Error	Description
Overrun Error	If the 16-byte receive FIFO overruns, the previously received byte is overwritten. The controller closes the buffer and sets both RxBD[OV] and USBER[RXB]. For **USB function mode** the NAK handshake is sent after the end of the received packet if the packet was received error-free.
Busy Error	A frame was received and discarded due to lack of buffers. The controller sets USBER[BSY].
Non Octet-Aligned Packet	If this error occurs, the controller writes the received data to the buffer, closes the buffer and sets both RxBD[NO] and USBER[RXB].
CRC Error	When a CRC error occurs, the controller closes the buffer, and sets both RxBD[CR] and USBER[RXB]. In isochronous mode (USEP*n*[TM] = 0b11), the USB controller reports a CRC error; however, there are no handshake packets (ACK) and the transfer continues normally when an error occurs.
Buffer Overflow	For **USB host mode** packet-level interface only. If the received data packet is larger than the allocated buffer, the remaining data is discarded, and TrBD[BOV] is set. The TXE1 interrupt bit is set.

27.9 USB Function Controller Initialization Example

The following is an example initialization sequence for the USB controller operating in function mode. It can be used to set up two function endpoints to fill transmit FIFOs so that data is ready for transmission when an IN token is received from the USB. The tokens can be generated using a USB traffic generator.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Clear FRAME_N.
4. Write (DPRAM+0x500) to EP1PTR, and (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
5. Write 0xBC80_0004 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of endpoint 1.
6. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of endpoint 1.
7. Write 0xBCC0_0004 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of endpoint 2.
8. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of endpoint 2.
9. Write 0xCAFE_CAFE to DPRAM+0x200 to set up the endpoint 1 Tx data pattern.
10. Write 0xFACE_FACE to DPRAM+0x210 to set up the endpoint 2 Tx data pattern.
11. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the endpoint 1 parameter RAM.
12. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 1 parameter RAM.
13. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the endpoint 1 parameter RAM.
14. Clear the TSTATE field of the endpoint 1 parameter RAM.
15. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
16. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
17. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
18. Clear the TSTATE field of the endpoint 2 parameter RAM.
19. Write 0x0000 to USEP1: Endpoint Number 0, control transfer, one packet only, and normal handshake.
20. Write 0x7200 to USEP2: Endpoint Number 7, bulk transfer, one packet only, and normal handshake.
21. Write 0x00 to the USMOD for full-speed 12 Mbps function endpoint mode and disable the USB.
22. Write 0x05 to the USAD for slave address 5.
23. Set USMOD[EN] to enable the USB controller.

24. Write 0x80 to USCOM to start filling the Tx FIFO with endpoint 1 data ready for transmission when an IN token is received.
25. Write 0x81 to USCOM to start filling the Tx FIFO with endpoint 2 data ready for transmission when an IN token is received.
26. Generate an IN token to address 5, endpoint number 0, control.
27. Generate an IN token to address 5, endpoint number 7, bulk.

27.10 Programming the USB Host Controller (Packet-Level)

The MPC8272 implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loop-back).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 27.10.1, "USB Host Controller Initialization Example.")

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Prepare tokens in separate BDs.
- Using software, append the CRC5 as part of the transmitted data because the CPM does not support automatic CRC5 generation.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TxBD[LSP] in the token's BD. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the data packet. After completion of the transaction, the host returns to full-speed operation. Note that LSP should be set only for token BDs.

27.10.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0000_0020 to DPRAM+0x500 to set up the RBASE and TBASE fields of the host endpoint parameter RAM.
5. Write 0x1818_0100 to DPRAM+0x504 to set up the RFCR, TFCR, and MRBLR fields of the host endpoint parameter RAM.

6. Write 0x0000_0020 to DPRAM+0x508 to set up the RBPTR and TBPTR fields of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.
9. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
10. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
11. Clear the TSTATE field of the endpoint 2 parameter RAM.
12. Write 0xB000_0000 to DPRAM+0x00 to set up the RxBD[Status and Control, Data Length] fields of the host endpoint.
13. Write DPRAM+0x100 to DPRAM+0x04 to set up the RxBD[Buffer Pointer] field of the host endpoint.
14. Write 0xB800_0003 to DPRAM+0x20 to set up the TxBD[Status and Control, Data Length] fields of the host endpoint.
15. Write DPRAM+0x200 to DPRAM+0x24 to set up the TxBD[Buffer Pointer] field of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0x698560 to DPRAM+0x200 to set up the host endpoint Tx data pattern. This pattern consists of the IN token and the CRC5.
19. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
20. Write 0x0020 to USEP1 for the host: non-isochronous transfer, multi-packet, packet-level interface.
21. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
22. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
23. Write 0x05 to the USAD for slave address 5.
24. Set USMOD[EN] to enable the USB controller.
25. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
26. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TxBD[Status and Control] of the host endpoint should contain 0x3800.
- TxBD[Data Length] of the host endpoint should contain 0x0003.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.

- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- RxBD[Status and Control] of the host endpoint should contain 0x3C00.
- RxBD[Data Length] of the host endpoint should contain 0x0005.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

27.11 Programming the USB Host Controller (Transaction-Level)

The MPC8272 implementation of a USB host uses the endpoint represented by USEP1 to control the host transmission and reception. The other endpoints are typically not used, except for testing purposes (loop-back).

Programming the USB controller to act as host is similar to configuring an endpoint for function operation. A general outline of how to program the host controller follows. (A more detailed example can be found in Section 27.11.1, “USB Host Controller Initialization Example.”)

- Set the host bit in the mode register (USBMOD[HOST] = 1) to configure the controller as a host.
- Set the multi-frame bit in the endpoint configuration register (USEP1[MF] = 1) to allow SETUP/OUT tokens and DATA0/DATA1 packets to be sent back-to-back.
- Set USEP1[RTE] to enable the transaction-level interface.
- Clock the USB host controller as a high speed function (48-MHz reference clock).
- For low-speed transactions with an external hub, set TrBD[LSP]. This causes the USB host controller to generate a preamble (PRE token) at full speed before changing the transmit rate to low speed and sending the token. After completion of the transaction, the host returns to full-speed operation.

27.11.1 USB Host Controller Initialization Example

The following is a local loopback example initialization sequence for the USB controller operating as a host. It can be used to set up the host endpoint and one function endpoint to demonstrate an IN token transaction.

1. Program CMXSCR to provide a 48 MHz clock to the USB controller.
2. Program the Port Registers to select USBRXD, USBRXP, USBRXN, USBTXP, USBTXN, and $\overline{\text{USBOE}}$.
3. Write (DPRAM+0x500) to EP1PTR, (DPRAM+0x520) to EP2PTR to set up the endpoint pointers.
4. Write 0x0020 to DPRAM+0x502 to set up the TBASE field of the host endpoint parameter RAM.
5. Write 0x1818 to DPRAM+0x504 to set up the RFCR and TFCR fields of the host endpoint parameter RAM.
6. Write 0x0020 to DPRAM+0x50a to set up the TBPTR field of the host endpoint parameter RAM.
7. Clear the TSTATE field of the host endpoint parameter RAM.
8. Initialize the HIMMR field of the host endpoint parameter RAM.
9. Write 0x0008_0028 to DPRAM+0x520 to set up the RBASE and TBASE fields of the endpoint 2 parameter RAM.

10. Write 0x1818_0100 to DPRAM+0x524 to set up the RFCR, TFCR, and MRBLR fields of the endpoint 2 parameter RAM.
11. Write 0x0008_0028 to DPRAM+0x528 to set up the RBPTR and TBPTR fields of the endpoint 2 parameter RAM.
12. Clear the TSTATE field of the endpoint 2 parameter RAM.
13. Write 0xB800_0040 to DPRAM+0x20 to set up the TrBD[Status and Control, Data Length] fields of the host endpoint.
14. Write DPRAM+0x100 to DPRAM+0x24 to set up the TrBD[Buffer Pointer] field of the host endpoint.
15. Write 0x8085 to DPRAM+0x28 to set up the TrBD token fields of the host endpoint.
16. Write 0xBC80_0003 to DPRAM+0x28 to set up the TxBD[Status and Control, Data Length] fields of the function endpoint.
17. Write DPRAM+0x210 to DPRAM+0x2C to set up the TxBD[Buffer Pointer] field of the function endpoint.
18. Write 0xABCD_1234 to DPRAM+0x210 to set up the function endpoint Tx data pattern.
19. Write 0x0030 to USEP1 for the host: non-isochronous transfer, multi-packet, transaction-level interface.
20. Write 0x1100 to USEP2 for the function: interrupt transfer, one packet only.
21. Write 0x06 to USMOD for full-speed 12 Mbps signaling, local loopback configuration (test and host modes set), and disable the USB.
22. Write 0x05 to the USAD for slave address 5.
23. Set USMOD[EN] to enable the USB controller.
24. Write 0x81 to the USCOM to start filling the Tx FIFO with endpoint 2 data to be ready for transmission when an IN token is received.
25. Write 0x80 to the USCOM to start transmitting the IN token.

The expected results are as follows:

- TrBD[Status and Control] of the host endpoint should contain 0x3800.
- TrBD[Data Length] of the host endpoint should contain 0x0005.
- TxBD[Status and Control] of endpoint 2 should contain 0x3C80.
- TxBD[Data Length] of endpoint 2 should contain 0x0003.
- The receive buffer of the host endpoint should contain 0xABCD_122B, 0x42xx_xxxx.

Chapter 28
Serial Management Controllers (SMCs)

The two serial management controllers (SMCs) are full-duplex ports that can be configured independently to support one of three protocols or modes—UART, transparent, or general-circuit interface (GCI). Simple UART operation is used to provide a debug/monitor port in an application, which frees the SCCs for other purposes. An SMC in UART mode is not as complex as that of an SCC in UART mode. The SMC clock can be derived from one of the internal baud rate generators or from an external clock signal. However, the clock should be a 16× clock.

In totally transparent mode, the SMC can be connected to a TDM channel (such as a T1 line) or directly to its own set of signals. The receive and transmit clocks are derived from the TDM channel, the internal baud rate generators, or from an external 1× clock. The transparent protocol allows the transmitter and receiver to use the external synchronization signal. An SMC in transparent mode is not as complex as that of an SCC in transparent mode.

Each SMC supports the C/I and monitor channels of the GCI bus, for which the SMC connects to a time-division multiplex (TDM) channel in a serial interface (SI2). SMCs support loopback and echo modes for testing. The SMC receiver and transmitter are double buffered, corresponding to an effective FIFO size (latency) of two characters. Chapter 14, "Serial Interface with Time-Slot Assigner," describes the GCI interface configuration.

Figure 28-1 shows the SMC block diagram.

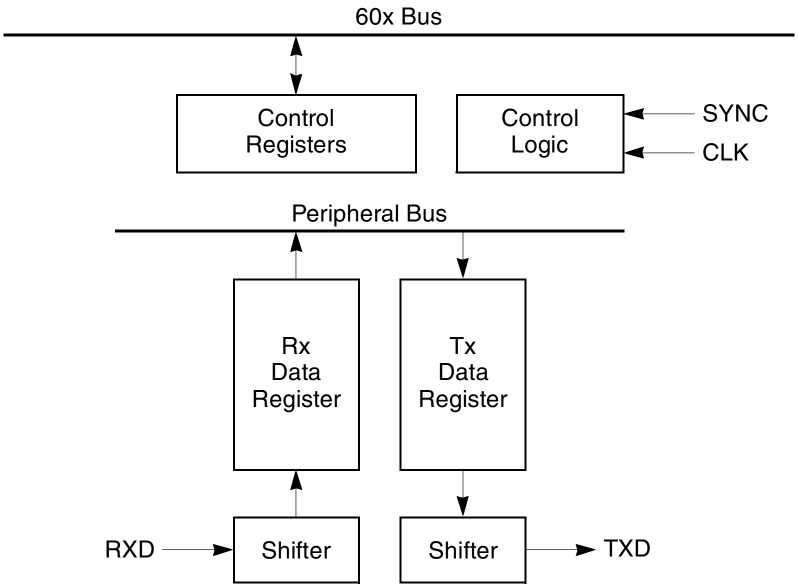


Figure 28-1. SMC Block Diagram

The receive data source can be L1RXD if the SMC is connected to a TDM channel of an SI2, or SMRXD if it is connected to the NMSI. The transmit data source can be L1TXD if the SMC is connected to a TDM, or SMTXD if it is connected to the NMSI.

If the SMC is connected to a TDM, the SMC receive and transmit clocks can be independent from each other, as defined in Chapter 14, “Serial Interface with Time-Slot Assigner.” However, if the SMC is connected to the NMSI, receive and transmit clocks must be connected to a single clock source (SMCLK), an internal signal name for a clock generated from the bank of clocks. SMCLK originates from an external signal or one of the four internal baud rate generators.

An SMC connected to a TDM derives a synchronization pulse from the TSA. An SMC connected to the NMSI using the transparent protocol can use $\overline{\text{SMSYN}}$ for synchronization to determine when to start a transfer. $\overline{\text{SMSYN}}$ is not used when the SMC is in UART mode.

28.1 Features

The following is a list of the SMC’s main features:

- Each SMC can implement the UART protocol on its own signals.
- Each SMC can implement a totally transparent protocol on a multiplexed or non-multiplexed line. This mode can also be used for a fast connection between PowerQUICC IIs.
- Each SMC channel fully supports the C/I and monitor channels of the GCI (IOM-2) in ISDN applications.
- Two SMCs support the two sets of C/I and monitor channels in the SCIT channels 0 and 1.
- Full-duplex operation
- Local loopback and echo capability for testing

28.2 Common SMC Settings and Configurations

The following sections describe settings and configurations common to the SMCs.

28.2.1 SMC Mode Registers (SMCMR1/SMCMR2)

The SMC mode registers (SMCMR1 and SMCMR2), shown in Figure 28-2, select the SMC mode as well as mode-specific parameters. The functions of SMCMR[8–15] are the same for each protocol. Bits 0–7 vary according to protocol selected by the SM bits.

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field: UART	—	CLEN				SL	PEN	PM	—		SM		DM		TEN	REN
Transparent						—	BS	REVD								
GCI						ME	—	C#								
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11A82 (SMCMR1), 0x11A92 (SMCMR2)															

Figure 28-2. SMC Mode Registers (SMCMR1/SMCMR2)

Table 28-1 describes SMCMR fields.

Table 28-1. SMCMR1/SMCMR2 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1–4	CLEN	Character length (UART). Number of bits in the character minus one. The total is the sum of 1 (start bit always present) + number of data bits (5–14) + number of parity bits (0 or 1) + number of stop bits (1 or 2). For example, for 8 data bits, no parity, and 1 stop bit, the total number of bits in the character is 1 + 8 + 0 + 1 = 10. So, CLEN should be programmed to 9. Characters range from 5–14 bits. If the data bit length is less than 8, the msbs of each byte in memory are not used on transmit and are written with zeros on receive. If the length is more than 8, the msbs of each 16-bit word are not used on transmit and are written with zeros on receive. The character must not exceed 16 bits. For a 14-bit data length, set SL to one stop bit and disable parity. For a 13-bit data length with parity enabled, set SL to one stop bit. Writing values 0 to 3 to CLEN causes erratic behavior.
		Character length (transparent). The values 3–15 specify 4–16 bits per character. If a character is less than 8 bits, the most-significant bits of the byte in buffer memory are not used on transmit and are written with zeros on receive. If character length is more than 8 bits but less than 16, the most-significant bits of the half-word in buffer memory are not used on transmit and are written with zeros on receive. Note: Using values 0–2 causes erratic behavior. Larger character lengths increase an SMC channel's potential performance and lowers the performance impact of other channels. For instance, using 16- rather than 8-bit characters is encouraged if 16-bit characters are acceptable in the end application.
		Character length (GCI). Number of bits in the C/I and monitor channels of the SCIT channels 0 or 1. (values 0–15 correspond to 1–16 bits) CLEN should be 13 for SCIT channel 0 or GCI (8 data bits, plus A and E bits, plus 4 C/I bits = 14 bits). It should be 15 for the SCIT channel 1 (8 data, bits, plus A and E bits, plus 6 C/I bits = 16 bits).
5	SL	Stop length. (UART) 0 One stop bit 1 Two stop bits
	—	Reserved, should be cleared (transparent)
	ME	Monitor enable. (GCI) 0 The SMC does not support the monitor channel. 1 The SMC supports the monitor channel.

Table 28-1. SMCMR1/SMCMR2 Field Descriptions (continued)

Bits	Name	Description
6	PEN	Parity enable. (UART) 0 No parity 1 Parity is enabled for the transmitter and receiver as determined by the PM bit.
	BS	Byte sequence (transparent). Controls the byte transmission sequence if REVD is set for a character length greater than 8 bits. Clear BS to maintain behavior compatibility with MC68360 QUICC. 0 Normal mode. This should be selected if the character length is not larger than 8 bits. 1 Transmit lower address byte first.
	—	Reserved, should be cleared (GCI)
7	PM	Parity mode. (UART) 0 Odd parity 1 Even parity.
	REVD	Reverse data. (transparent) 0 Normal mode 1 Reverse the character bit order. The msb is sent first.
	C#	SCIT channel number. (GCI) 0 SCIT channel 0 1 SCIT channel 1. Required for Siemens ARCOFI and SGS S/T chips.
8–9	—	Reserved, should be cleared
10–11	SM	SMC mode. 00 GCI or SCIT support 01 Reserved 10 UART (must be selected for SMC UART operation) 11 Totally transparent operation
12–13	DM	Diagnostic mode. 00 Normal operation 01 Local loopback mode 10 Echo mode 11 Reserved
14	TEN	SMC transmit enable. 0 SMC transmitter disabled 1 SMC transmitter enabled
15	REN	SMC receive enable. 0 SMC receiver disabled 1 SMC receiver enabled

28.2.2 SMC Buffer Descriptor Operation

In UART and transparent modes, the SMC’s memory structure is like that of the SCC, except that SMC-associated data is stored in buffers. Each buffer is referenced by a BD and organized in a BD table located in the dual-port RAM. See Figure 28-3.

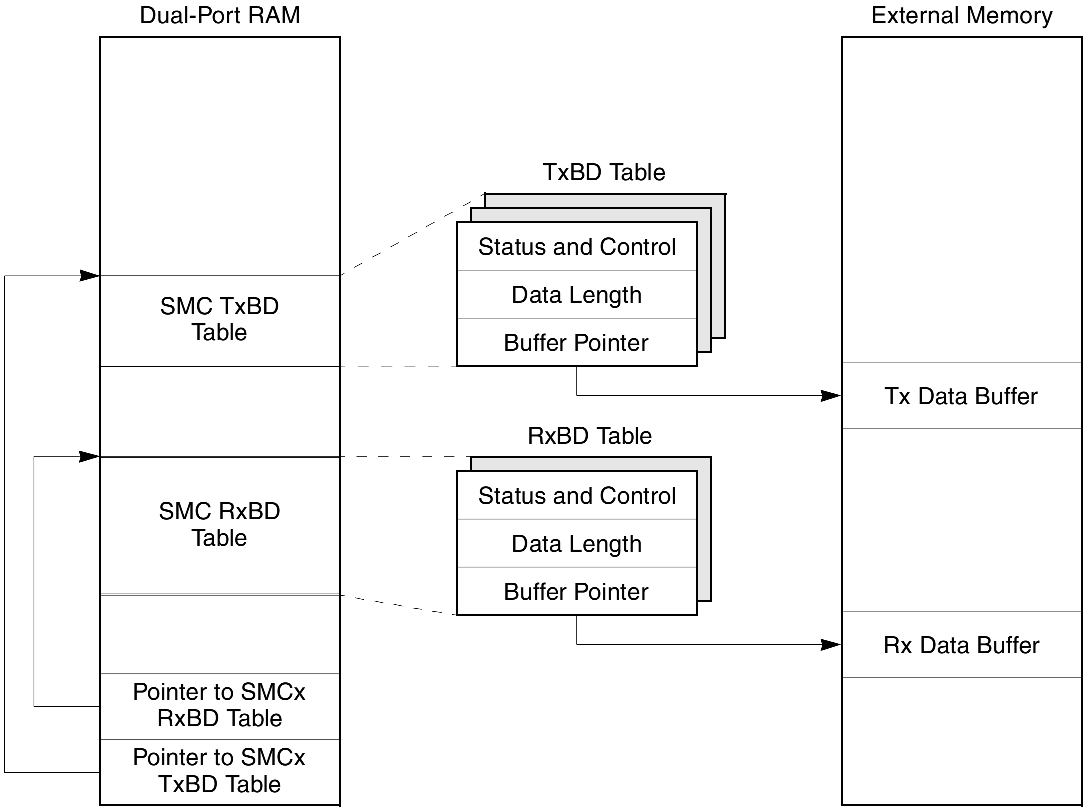


Figure 28-3. SMC Memory Structure

The BD table allows buffers to be defined for transmission and reception. Each table forms a circular queue. The CP uses BDs to confirm reception and transmission so that the processor knows buffers have been serviced. The data resides in external or internal buffers.

When SMCs are configured to operate in GCI mode, their memory structure is predefined to be one half word long for transmit and one half word long for receive. For more information on these half-word structures, see Section 28.5, "The SMC in GCI Mode."

28.2.3 SMC Parameter RAM

The CP accesses each SMC's parameter table using a user-programmed pointer (SMC*x*_BASE) located in the parameter RAM; see Section 13.5.2, "Parameter RAM." Each SMC parameter RAM table can be placed at any 64-byte aligned address in the dual-port RAM's general-purpose area (banks 1–8, 11 and 12). The protocol-specific portions of the SMC parameter RAM are discussed in the sections that follow. The SMC parameter RAM shared by the UART and transparent protocols is shown in Table 28-2. Parameter RAM for GCI protocol is described in Table 28-17.

Table 28-2. SMC UART and Transparent Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	RxBDs and TxBDs base address. (BD table pointer) Define starting points in the dual-port RAM of the set of BDs for the SMC send and receive functions. They allow flexible partitioning of the BDs. By selecting RBASE and TBASE entries for all SMCs and by setting W in the last BD in each list, BDs are allocated for the send and receive side of every SMC. Initialize these entries before enabling the corresponding channel. Configuring BD tables of two enabled SMCs to overlap causes erratic operation. RBASE and TBASE should be a multiple of eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code. See Section 28.2.3.1, “SMC Function Code Registers (RFCR/TFCR).”
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. The most bytes the PowerQUICC II writes to a receive buffer before moving to the next buffer. It can write fewer bytes than MRBLR if a condition like an error or end-of-frame occurs, but it cannot exceed MRBLR. PowerQUICC II buffers should not be smaller than MRBLR. SMC transmit buffers are unaffected by MRBLR. Transmit buffers can be individually given varying lengths through the data length field. MRBLR can be changed while an SMC is operating only if it is done in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). This occurs when the CP shifts control to the next RxBD, so the change does not take effect immediately. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the SMC receiver is disabled. MRBLR should be greater than zero and should be even if character length exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state. [2] Can be used only by the CP.
0x0C	—	Word	Rx internal data pointer. [2] Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the next BD for each SMC channel that the receiver transfers data to when it is in idle state, or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP initializes RBPTR to the value in RBASE. Most applications never need to write RBPTR, but it can be written when the receiver is disabled or when no receive buffer is in use.
0x12	—	Hword	Rx internal byte count. [2] A down-count value initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	—	Word	Rx temp [2] Can be used only by the CP.
0x18	TSTATE	Word	Tx internal state. [2] Can be used only by the CP.
0x1C	—	Word	Tx internal data pointer. [2] Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the next BD for each SMC channel the transmitter transfers data from when it is in idle state or to the current BD during frame transmission. After reset or when the end of the table is reached, the CP initializes TBPTR to the TBASE value. Most applications never need to write TBPTR, but it can be written when the transmitter is disabled or when no transmit buffer is in use. For instance, after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission.
0x22	—	Hword	Tx internal byte count. [2] A down-count value initialized with the TxBD data length and decremented with every byte the SDMA channels read.

Table 28-2. SMC UART and Transparent Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x24	—	Word	Tx temp. [2] Can be used only by the CP.
0x28	**MAX_IDL**	Hword	Maximum idle characters. (UART protocol-specific parameter) When a character is received on the line, the SMC starts counting idle characters received. If MAX_IDL idle characters arrive before the next character, an idle time-out occurs and the buffer closes, which sends an interrupt request to the core to receive data from the buffer. MAX_IDL demarcates frames in UART mode. Clearing MAX_IDL disables the function so the buffer never closes, regardless of how many idle characters are received. An idle character is calculated as follows: 1 + data length (5 to 14) + 1 (if parity bit is used) + number of stop bits (1 or 2). For example, for 8 data bits, no parity, and 1 stop bit, character length is 10 bits.
0x2A	IDLC	Hword	Temporary idle counter. (UART protocol-specific parameter) Down-counter in which the CP stores the current idle counter value in the MAX_IDL time-out process.
0x2C	BRKLN	Hword	Last received break length. (UART protocol-specific parameter) Holds the length of the last received break character sequence measured in character units. For example, if the receive signal is low for 20 bit times and the defined character length is 10 bits, BRKLN = 0x002, indicating that the break sequence is at least 2 characters long. BRKLN is accurate to within one character length.
0x2E	**BRKEC**	Hword	Receive break condition counter. (UART protocol-specific parameter) Counts break conditions on the line. A break condition may last for hundreds of bit times, yet BRKEC increments only once during that period.
0x30	**BRKCR**	Hword	Break count register (transmit). (UART protocol-specific parameter) Determines the number of break characters the UART controller sends. Set when the SMC sends a break character sequence after a STOP TRANSMIT command. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character is 10 zeros.
0x32	R_MASK	Hword	Temporary bit mask. (UART protocol-specific parameter)
0x34	—	Word	SDMA temp

[1] From the pointer value programmed in SMC*x*_BASE: SMC1_BASE at 0x87FC, SMC2_BASE at IMMR + 0x88FC.

[2] Not accessed for normal operation. May hold helpful information for experienced users and for debugging.

To extract data from a partially full receive buffer, issue a CLOSE RXBD command.

Certain parameter RAM values must be initialized before the SMC is enabled. Other values are initialized or written by the CP. Once values are initialized, software typically does not need to update them because activity centers mostly around transmit and receive BDs rather than parameter RAM. However, note the following:

- Parameter RAM can be read at any time.
- Values that pertain to the SMC transmitter can be written only if SMCMR[TEN] is zero or between the STOP TRANSMIT and RESTART TRANSMIT commands.
- Values for the SMC receiver can be written only when SMCMR[REN] is zero, or, if the receiver is previously enabled, after an ENTER HUNT MODE command is issued but before the CLOSE RXBD command is issued and REN is set.

28.2.3.1 SMC Function Code Registers (RFCR/TFCR)

The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 28-4 shows the register format.

	0	1	2	3	4	5	6	7
Field			**GBL**	**BO**		**TC2**	**DTB**	—
R/W	R/W							
Addr	SMC base + 0x04 (RFCR)/SMC base + 0x05 (TFCR)							

Figure 28-4. SMC Function Code Registers (RFCR/TFCR)

Table 28-3 describes FCR fields.

Table 28-3. RFCR/TFCR Field Descriptions

Bit	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global access bit 0 Disable memory snooping 1 Enable memory snooping
3–4	**BO**	Byte ordering. Selects byte ordering of the data buffer. 00 The DEC/Intel convention (swapped operation or little-endian). The transmission order of bytes within a buffer word is opposite of Freescale mode. (32-bit port size memory only). 01 Munged little-endian. As data is sent onto the serial line from the buffer, the LSB of the buffer double word contains data to be sent earlier than the MSB of the same double word. 1x Freescale (big-endian) byte ordering (normal operation). As data is sent onto the serial line from the buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same word.
5	**TC2**	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator. 0 Use 60x bus for SDMA operation 1 Reserved
7	—	Reserved, should be cleared

28.2.4 Disabling SMCs On-the-Fly

An SMC can be disabled and reenabled later by ensuring that buffers are closed properly and new data is transferred to or from a new buffer. Such a sequence is required if the parameters to be changed are not dynamic. If the register or bit description states that dynamic changes are allowed, the sequences need not be followed and the register or bits may be changed immediately.

NOTE

The SMC does not have to be fully disabled for parameter RAM to be modified. Table 28-2 describes when parameter RAM values can be modified. To disable all SCCs, SMCs, the SPI, and the I^{2}C, use the CPCR to reset the CPM with a single command.

28.2.4.1 SMC Transmitter Full Sequence

Follow these steps to fully enable or disable the SMC transmitter:

1. If the SMC is sending data, issue a STOP TRANSMIT command to stop transmission smoothly. If the SMC is not sending, if TBPTR is overwritten, or if an INIT TX PARAMETERS command is executed, this command is not required.
2. Clear SMCMR[TEN] to disable the SMC transmitter and put it in reset state.
3. Update SMC transmit parameters, including the parameter RAM. To switch protocols or reinitialize parameters, issue an INIT TX PARAMETERS command.
4. Issue a RESTART TRANSMIT if an INIT TX PARAMETERS was issued in step 3.
5. Set SMCMR[TEN]. Transmission now begins using the TxBD that the TBPTR value pointed to as soon as the R bit is set in the TxBD.

28.2.4.2 SMC Transmitter Shortcut Sequence

This shorter sequence reinitializes transmit parameters to the state they had after reset.

1. Clear SMCMR[TEN].
2. Issue an INIT TX PARAMETERS command and make any additional changes.
3. Set SMCMR[TEN].

28.2.4.3 SMC Receiver Full Sequence

Follow these steps to fully enable or disable the receiver:

1. Clear SMCMR[REN]. Reception is aborted immediately, which disables the SMC receiver and puts it in a reset state.
2. Modify SMC receive parameters, including parameter RAM. To switch protocols or reinitialize SMC receive parameters, issue an INIT RX PARAMETERS command.
3. Issue a CLOSE RXBD command if INIT RX PARAMETERS was not issued in step 2.
4. Set SMCMR[REN]. Reception immediately uses the RxBD that RBPTR pointed to if E is set in the RxBD.

28.2.4.4 SMC Receiver Shortcut Sequence

This shorter sequence reinitializes receive parameters to their state after reset.

1. Clear SMCMR[REN].
2. Issue an INIT RX PARAMETERS command and make any additional changes.
3. Set SMCMR[REN].

28.2.4.5 Switching Protocols

To switch the protocol that the SMC is executing without resetting the board or affecting any other SMC, use one command and follow these steps:

1. Clear SMCMR[REN] and SMCMR[TEN].

2. Issue an INIT TX AND RX PARAMETERS COMMAND to initialize transmit and receive parameters. Make any additional SMCMR changes.
3. Set SMCMR[REN, TEN]. The SMC is now enabled with the new protocol.

28.2.5 Saving Power

When SMCMR[TEN, REN] are cleared, the SMC consumes little power.

28.2.6 Handling Interrupts in the SMC

Follow these steps to handle an interrupt in the SMC:

1. Once an interrupt occurs, read SMCE to identify the interrupt source. The SMCE bits are usually cleared at this time.
2. Process the TxBD to reuse it if SMCE[TXB] is set. Extract data from the RxBD if SMCE[RXB] is set. To send another buffer, set TxBD[R].
3. Execute the **rfi** instruction.

28.3 SMC in UART Mode

SMCs generally offer less functionality and performance in UART mode than do SCCs, which makes them more suitable for simpler debug/monitor ports instead of full-featured UARTs. SMCs do not support the following features in UART mode:

- $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ signals
- Receive and transmit sections clocked at different rates
- Fractional stop bits
- Built-in multidrop modes
- Freeze mode for implementing flow control
- Isochronous (1× clock) operation (a 16× clock is required for UART operation.)
- Interrupts on special control character reception
- Ability to transmit data on demand using the TODR
- SCCS register to determine idle status of the receive signal
- Other features for the SCCs as described in the GSMR

However, SMCs allow a data length of up to 14 bits; SCCs only support up to 8 bits.

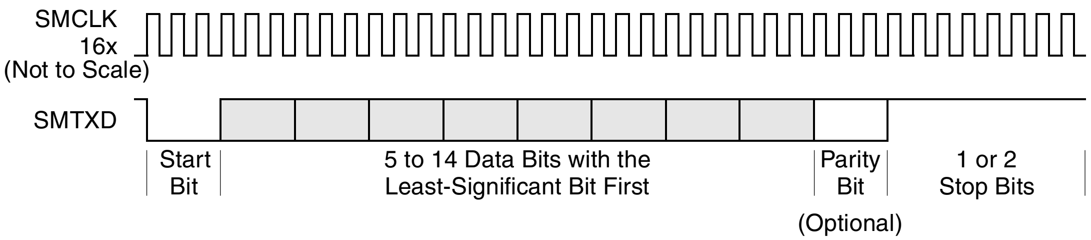


Figure 28-5. SMC UART Frame Format

28.3.1 Features

The following list summarizes the main features of the SMC in UART mode:

- Flexible message-oriented data structure
- Programmable data length (5–14 bits)
- Programmable 1 or 2 stop bits
- Even/odd/no parity generation and checking
- Frame error, break, and idle detection
- Transmit preamble and break sequences
- Received break character length indication
- Continuous receive and transmit modes

28.3.2 SMC UART Channel Transmission Process

The UART transmitter is designed to work with almost no intervention from the core. When the core enables the SMC transmitter, it starts sending idles. The SMC immediately polls the first BD in the transmit channel BD table and once every character time after that, depending on character length. When there is a message to transmit, the SMC fetches data from memory and starts sending the message.

When a BD data is completely written to the transmit FIFO, the SMC writes the message status bits into the BD and clears R. An interrupt is issued if the I bit in the BD is set. If the next TxBD is ready, the data from its buffer is appended to the previous data and sent over the transmit signal without any gaps between buffers. If the next TxBD is not ready, the SMC starts sending idles and waits for the next TxBD to be ready.

By appropriately setting the I bit in each BD, interrupts can be generated after each buffer, a specific buffer, or each block is sent. The SMC then proceeds to the next BD. If the CM bit is set in the TxBD, the R bit is not cleared, allowing a buffer to be automatically resent next time the CP accesses this buffer. For instance, if a single TxBD is initialized with the CM and W bits set, the buffer is sent continuously until R is cleared in the BD.

28.3.3 SMC UART Channel Reception Process

When the core enables the SMC receiver, it enters hunt mode and waits for the first character. The CP then checks the first RxBD to see if it is empty and starts storing characters in the buffer. When the buffer is full or the MAX_IDL timer expires (if enabled), the SMC clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the buffer's length, the SMC fetches the next BD, and, if it is empty, continues transferring data to this BD's buffer. If CM is set in the RxBD, the E bit is not cleared, so the CP can overwrite this buffer on its next access.

28.3.4 Programming the SMC UART Controller

UART mode is selected by setting SMCMR[SM] to 0b10. See Section 28.2.1, "SMC Mode Registers (SMCMR1/SMCMR2)." UART mode uses the same data structure as other modes. This structure supports multibuffer operation and allows break and preamble sequences to be sent. Overrun, parity, and framing errors are reported through the BDs. At its simplest, the SMC UART controller functions in a character-oriented environment, whereas each character is sent with the selected stop bits and parity. They are received into separate 1-byte buffers. A maskable interrupt can be generated when each buffer is received.

Many applications can take advantage of the message-oriented capabilities that the SMC UART supports through linked buffers for sending or receiving. Data is handled in a message-oriented environment, so entire messages can be handled instead of individual characters. A message can span several linked buffers; each one can be sent and received as a linked list of buffers without core intervention, which simplifies programming and saves processor overhead. In a message-oriented environment, an idle sequence is used as the message delimiter. The transmitter can generate an idle sequence before starting a new message, and the receiver can close a buffer when an idle sequence is found.

28.3.5 SMC UART Transmit and Receive Commands

Table 28-4 describes transmit commands issued to the CPCR.

Table 28-4. Transmit Commands

Command	Description
STOP TRANSMIT	Disables transmission of characters on the transmit channel. If the SMC UART controller receives this command while sending a message, it stops sending. The SMC UART controller finishes sending any data that has already been sent to its FIFO and shift register and then stops sending data. The TBPTR is not advanced when this command is issued. The SMC UART controller sends a programmable number of break sequences and then sends idles. The number of break sequences, which can be zero, should be written to the BRKCR before this command is issued to the SMC UART controller.
RESTART TRANSMIT	Enables characters to be sent on the transmit channel. The SMC UART controller expects it after disabling the channel in its SMCMR and after issuing the STOP TRANSMIT command. The SMC UART controller resumes transmission from the current TBPTR in the channel's TxBD table.
INIT TX PARAMETERS	Initializes transmit parameters in this serial channel's parameter RAM to their reset state and should only be issued when the transmitter is disabled. The INIT TX and RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Table 28-5 describes receive commands issued to the CPCR.

Table 28-5. Receive Commands

Command	Description
ENTER HUNT MODE	Use the CLOSE RXBD command instead ENTER HUNT MODE for an SMC UART channel.
CLOSE RXBD	Forces the SMC to close the current receive BD if it is currently being used and to use the next BD in the list for any subsequently received data. If the SMC is not receiving data, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel parameter RAM to reset state. Issue it only if the receiver is disabled. INIT TX AND RX PARAMETERS resets both receive and transmit parameters.

28.3.6 Sending a Break

A break is an all-zeros character without stop bits. It is sent by issuing a STOP TRANSMIT command. After sending any outstanding data, the SMC sends a character of consecutive zeros, the number of which is the sum of the character length and the number of start, parity, and stop bits. The SMC sends a programmable number of break characters according to BRKCR and reverts to idle or sends data if a RESTART TRANSMIT is issued before completion. When the break completes, the transmitter sends at least one idle character before sending any data to guarantee recognition of a valid start bit.

28.3.7 Sending a Preamble

A preamble sequence provides a way to ensure that the line is idle before a new message transfer begins. The length of the preamble sequence is constructed of consecutive ones that are one-character long. If the preamble bit in a BD is set, the SMC sends a preamble sequence before sending that buffer. For 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones would be sent before the first character in the buffer. If no preamble sequence is sent, data from two ready transmit buffers can be sent on the transmit signal with no delay between them.

28.3.8 Handling Errors in the SMC UART Controller

The SMC UART controller reports character reception error conditions through the channel BDs and the SMCE. Table 28-6 describes the reception errors. The SMC UART controller has no transmission errors.

Table 28-6. SMC UART Errors

Error	Description
Overrun	The SMC maintains a two-character length FIFO for receiving data. Data is moved to the buffer after the first character is received into the FIFO; if a receiver FIFO overrun occurs, the channel writes the received character into the internal FIFO. It then writes the character to the buffer, closes it, sets the OV bit in the BD, and generates the RXB interrupt if it is enabled. Reception then resumes as normal. Overrun errors that occasionally occur when the line is idle can be ignored.
Parity	The channel writes the received character to the buffer, closes it, sets the PR bit in the BD, and generates the RXB interrupt if it is enabled. Reception then resumes as normal.
Idle sequence receive	An idle is found when a character of all ones is received, at which point the channel counts consecutive idle characters. If the count reaches MAX_IDL, the buffer is closed and an RXB interrupt is generated. If no receive buffer is open, this does not generate an interrupt or any status information. The idle counter is reset each time a character is received.
Framing	The SMC received a character with no stop bit. When it occurs, the channel writes the received character to the buffer, closes the buffer, sets FR in the BD, and generates the RXB interrupt if it is enabled. When this error occurs, parity is not checked for the character.
Break sequence	The SMC receiver received an all-zero character with a framing error. The channel increments BRKEC, generates a maskable BRK interrupt in SMCE, measures the length of the break sequence, and stores this value in BRKLN. If the channel was processing a buffer when the break was received, the buffer is closed with the BR bit in the RxBD set. The RXB interrupt is generated if it is enabled.

28.3.9 SMC UART RxBD

Using the BDs, the CP reports information about the received data on a per-buffer basis. It then closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- An error is received during message processing.

- A full receive buffer is detected.
- A programmable number of consecutive idle characters are received.

Figure 28-6 shows the format of the SMC UART RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	—		**CM**	ID	—		BR	FR	PR	—	OV	—
Offset + 2	**Data Length**															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 28-6. SMC UART RxBD

Table 28-7 describes RxBD fields.

Table 28-7. SMC UART RxBD Field Descriptions

Bit	Name	Description
0	**E**	Empty. 0 The buffer is full or data reception stopped due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E is zero. 1 The buffer is empty or reception is in progress. This RxBD and its buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (last BD in RxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 The SMCE[RXB] is set when this buffer is completely filled by the CP, indicating the need for the core to process the buffer. RXB can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear the E bit after this BD is closed, allowing the CP to automatically overwrite the buffer when it next accesses the BD. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7	ID	Buffer closed on reception of idles. Set when the buffer has closed because a programmable number of consecutive idle sequences is received. The CP writes ID after received data is in the buffer.
8–9	—	Reserved, should be cleared
10	BR	Buffer closed on reception of break. Set when the buffer closes because a break sequence was received. The CP writes BR after the received data is in the buffer.
11	FR	Framing error. Set when a character with a framing error is received and located in the last byte of this buffer. A framing error is a character with no stop bit. A new receive buffer is used to receive additional data. The CP writes FR after the received data is in the buffer.

Table 28-7. SMC UART RxBD Field Descriptions (continued)

Bit	Name	Description
12	PR	Parity error. Set when a character with a parity error is received in the last byte of the buffer. A new buffer is used for additional data. The CP writes PR after received data is in the buffer.
13	—	Reserved, should be cleared
14	OV	Overrun. Set when a receiver overrun occurs during reception. The CP writes OV after the received data is in the buffer.
15	—	Reserved, should be cleared

Data length represents the number of octets the CP writes into the buffer. After data is received in the buffer, the CP only writes them once as the BD closes. Note that the memory allocated for this buffer should be no smaller than MRBLR. The Rx data buffer pointer points to the first location of the buffer and must be even. The buffer can be in internal or external memory. Figure 28-7 shows the UART RxBD process, showing RxBDs after they receive 10 characters, an idle period, and five characters (one with a framing error). The example assumes that MRBLR = 8.

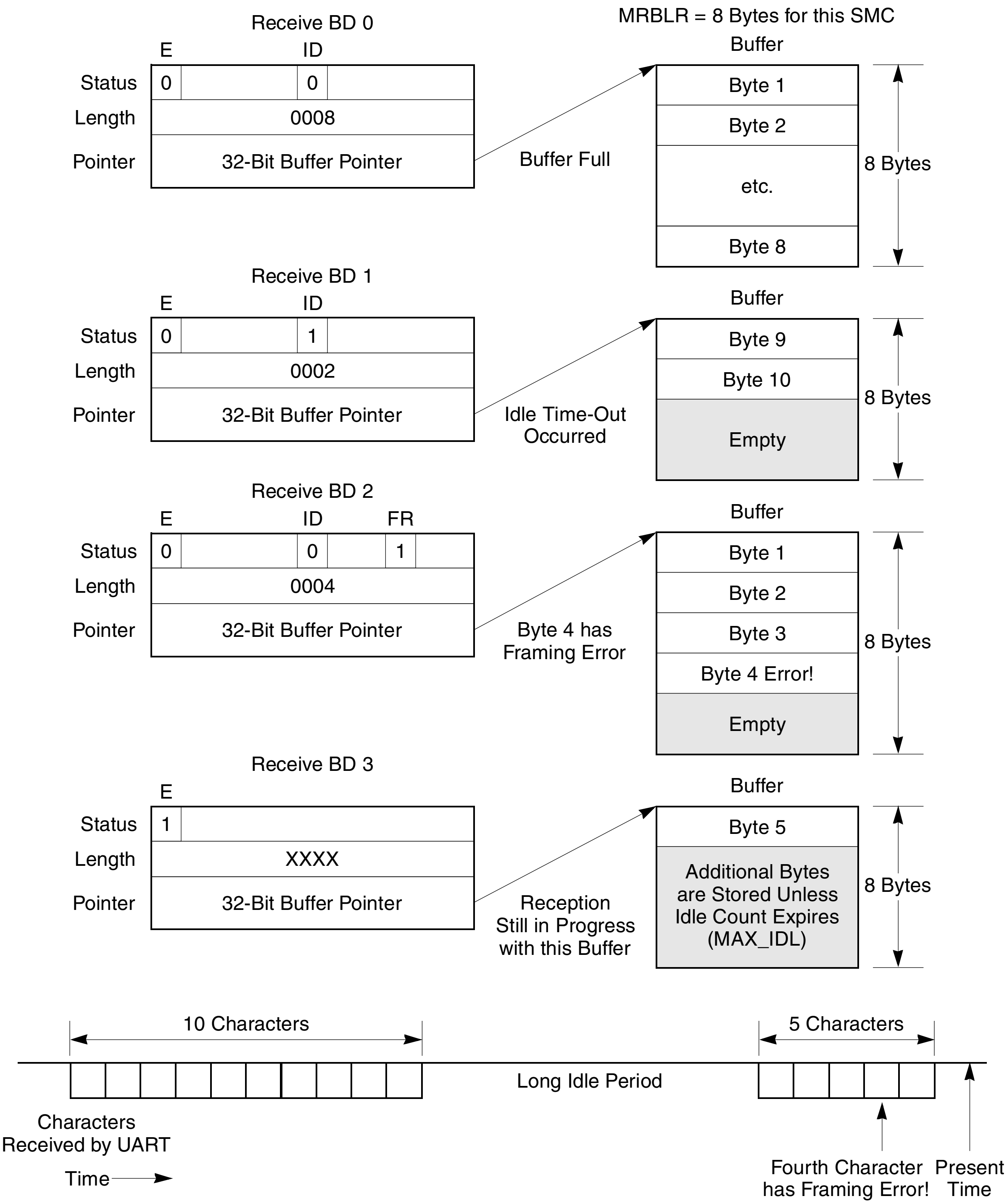


Figure 28-7. RxBD Example

28.3.10 SMC UART TxBD

Data is sent to the CP for transmission on an SMC channel by arranging it in buffers referenced by the channel TxBD table. Using the BDs, the CP confirms transmission or indicates error conditions so that the processor knows the buffers have been serviced. An SMC UART TxBD is displayed in Figure 28-8.

	0	1	2	3	4	5	6	7	8	15
Offset + 0	R	—	W	I	—		CM	P	—	
Offset + 2	Data Length									
Offset + 4	Tx Data Buffer Pointer									
Offset + 6										

Figure 28-8. SMC UART TxBD

Table 28-8 describes SMC UART TxBD fields.

Table 28-8. SMC UART TxBD Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer is not ready for transmission; BD and its buffer can be altered. The CP clears R after the buffer has been sent or an error occurs. 1 The buffer has not been completely sent. This BD cannot updated while R is set.
1	—	Reserved, should be cleared
2	W	Wrap (final BD in the TxBD table) 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 The SMCE[TXB] is set when this buffer is serviced. TXB can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	CM	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed and automatically retransmits the buffer when it accesses this BD next.
7	P	Preamble 0 No preamble sequence is sent. 1 The UART sends one all-ones character before it sends the data so that the other end detects an idle line before the data is received. If this bit is set and the data length of this BD is zero, only a preamble is sent.
8–15	—	Reserved, should be cleared

Data length represents the number of octets that the CP should transmit from this BD data buffer. However, it is never modified by the CP and normally is greater than zero. It can be zero if P is set and only a preamble is sent. If there are more than 8 bits in the UART character, data length should be even. For example, to transmit three UART characters of 8-bit data, 1 start, and 1 stop, initialize the data length field to 3. To send three UART characters of 9-bit data, 1 start, and 1 stop, the data length field should 6, because the three 9-bit data fields occupy three half words in memory (the 9 least-significant bits of each half word).

Tx data buffer pointer points to the first location of the buffer. It can be even or odd, unless the number of data bits in the UART character is greater than 8 bits. Then the buffer pointer must be even. For instance, the pointer to 8-bit data, 1 start, and 1 stop characters can be even or odd, but the pointer to 9-bit data, 1 start, and 1 stop characters must be even. The buffer can reside in internal or external memory.

28.3.11 SMC UART Event Register (SMCE)/Mask Register (SMCM)

The SMC event register (SMCE) generates interrupts and report events recognized by the SMC UART channel. When an event is recognized, the SMC UART controller sets the corresponding SMCE bit. Bits are cleared by writing a 1; writing 0 has no effect. The SMC mask register (SMCM) has the same bit format as SMCE. Setting an SMCM bit enables, and clearing it disables, the corresponding interrupt. All unmasked bits must be cleared before the CP clears the internal interrupt request. Figure 28-9 represents the SMCE/SMCM registers.

	0	1	2	3	4	5	6	7
Field	—	BRKE	—	BRK	—	BSY	TXB	RXB
Reset	0							
R/W	R/W							
Addr	0x11A86 (SMCE1), 0x11A96 (SMCE2)/ 0x11A8A (SMCM1), 0x11A9A (SMCM2)							

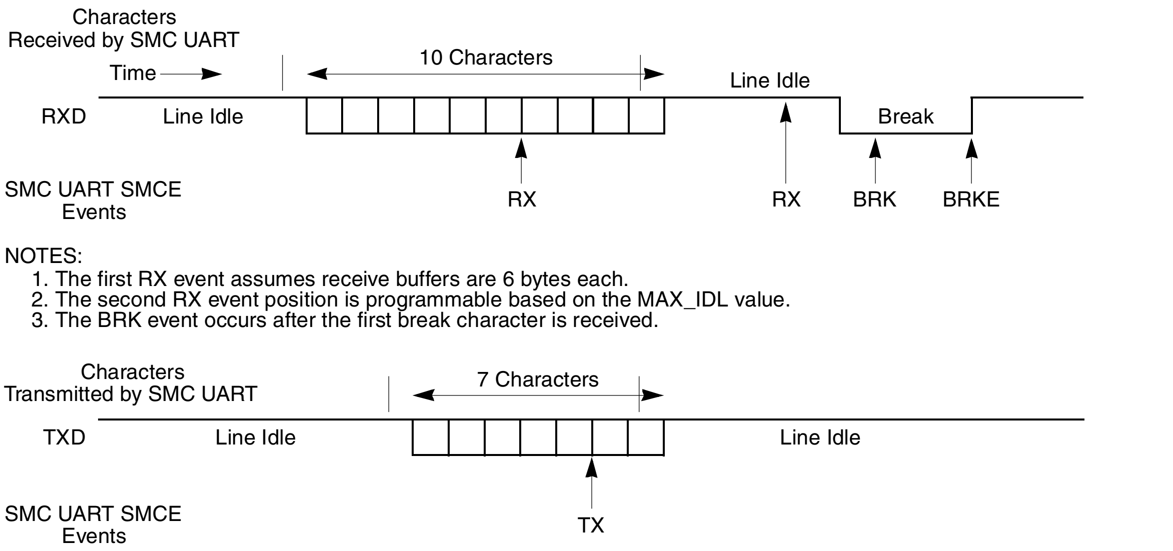
Figure 28-9. SMC UART Event Register (SMCE)/Mask Register (SMCM)

Table 28-9 describes SMCE/SMCM fields.

Table 28-9. SMCE/SMCM Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1	BRKE	Break end. Set no sooner than after one idle bit is received after the break sequence.
2	—	Reserved, should be cleared
3	BRK	Break character received. Set when a break character is received. If a very long break sequence occurs, this interrupt occurs only once after the first all-zeros character is received.
4	—	Reserved, should be cleared
5	BSY	Busy condition. Set when a character is received and discarded due to a lack of buffers. Set no sooner than the middle of the last stop bit of the first receive character for which there is no available buffer. Reception resumes when an empty buffer is provided.
6	TXB	Tx buffer. Set when the transmit data of the last character in the buffer is written to the transmit FIFO. Wait two character times to ensure that data is completely sent over the transmit signal.
7	RXB	Rx buffer. Set when a buffer is received and its associated RxBD is closed. Set no sooner than the middle of the last stop bit of the last character that is written to the receive buffer.

Figure 28-10 shows an example of the timing of various events in the SMCE.



NOTES:
1. The first RX event assumes receive buffers are 6 bytes each.
2. The second RX event position is programmable based on the MAX_IDL value.
3. The BRK event occurs after the first break character is received.

NOTES:
1. The TX event assumes all seven characters were put into a single buffer, and the TX event occurred when the seventh character was written to the SMC transmit FIFO.

Figure 28-10. SMC UART Interrupts Example

28.3.12 SMC UART Controller Programming Example

The following initialization sequence assumes 9,600 baud, 8 data bits, no parity, and 1 stop bit in a 66-MHz system. BRG1 and SMC1 are used. (The SMC transparent programming example uses an external clock configuration; see Section 28.4.11, “SMC Transparent NMSI Programming Example.”)

1. Configure the port D pins to enable SMTXD1 and SMRXD1. Set PPARD[8,9] and PDIRD[9]. Clear PDIRD[8] and PSORD[8,9].
2. Configure the BRG1. Write BRGC1 with 0x0001_035A. The DIV16 bit is not used and the divider is 429 (decimal). The resulting BRG1 clock is 16× the preferred bit rate.
3. Connect BRG1 to SMC1 using the CPM mux by clearing CMXSMR[SMC1, SMC1CS].
4. In address 0x87FC, assign a pointer to the SMC1 parameter RAM.
5. Assuming one RxBD at the beginning of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
6. Write 0x1D01_0000 to CPCR to execute the INIT RX AND TX PARAMETERS command.
7. Write RFCR and TFCR with 0x10 for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. Assume 16 bytes, so MRBLR = 0x0010.
9. Write MAX_IDL with 0x0000 in the SMC UART-specific parameter RAM to disable the MAX_IDL functionality for this example.
10. Clear BRKLN and BRKEC in the SMC UART-specific parameter RAM.
11. Set BRKCR to 0x0001; if a STOP TRANSMIT COMMAND is issued, one break character is sent.

12. Initialize the RxBD. Assume the Rx data buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (not required), and 0x0000_1000 to RxBD[Buffer Pointer].
13. Assuming the Tx data buffer is at 0x0000_2000 in main memory and contains five 8-bit characters, write 0xB000 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
14. Write 0xFF to the SMCE1 register to clear any previous events.
15. Write 0x57 to the SMCM1 register to enable all possible SMC1 interrupts.
16. Write 0x0000_1000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Write 0xFFFF_FFFF to the SIU interrupt pending register low (SIPNR_L) to clear events.
17. Write 0x4820 to SMCMR to configure normal operation (not loopback), 8-bit characters, no parity, 1 stop bit. The transmitter and receiver are not yet enabled.
18. Write 0x4823 to SMCMR to enable the SMC transmitter and receiver. This additional write ensures that the TEN and REN bits are enabled last.

After 5 bytes are sent, the TxBD is closed. The receive buffer closes after receiving 16 bytes. Subsequent data causes a busy (out-of-buffers) condition since only one RxBD is ready.

28.4 SMC in Transparent Mode

Compared to the SCC in transparent mode, the SMCs generally offer less functionality, which helps them provide simpler functions and slower speeds. Transparent mode is selected by programming SMCMR[SM] to 0b10. Section 28.2.1, “SMC Mode Registers (SMCMR1/SMCMR2),” describes other protocol-specific bits in the SMCMR. The SMC in transparent mode does not support the following features:

- Independent transmit and receive clocks, unless connected to a TDM channel of SI2
- CRC generation and checking
- Full $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ signals (supports only one $\overline{\text{SMSYN}}$ signal)
- Ability to transmit data on demand using the TODR
- Receiver/transmitter in transparent mode while executing another protocol
- 4-, 8-, or 16-bit SYNC recognition
- Internal DPLL support

However, the SMC in transparent mode provides a data character length option of 4 to 16 bits, whereas the SCCs provide 8 or 32 bits, depending on GSMR[RFW]. The SMC in transparent mode is also referred to as the SMC transparent controller.

28.4.1 Features

The following list summarizes the features of the SMC in transparent mode:

- Flexible data buffers
- Connects to a TDM bus using the TSA in SI2

- Transmits and receives transparently on its own set of signals using a sync signal to synchronize the beginning of transmission and reception to an external event
- Programmable character length (4–16)
- Reverse data mode
- Continuous transmission and reception modes
- Four commands

28.4.2 SMC Transparent Channel Transmission Process

The transparent transmitter is designed to work with almost no core intervention. When the core enables the SMC transmitter in transparent mode, it starts sending idles. The SMC immediately polls the first BD in the transmit channel BD table and once every character time, depending on the character length (every 4 to 16 serial clocks). When there is a message to transmit, the SMC fetches the data from memory and starts sending the message when synchronization is achieved.

Synchronization can be achieved in two ways. First, when the transmitter is connected to a TDM channel, it can be synchronized to a time slot. Once the frame sync is received, the transmitter waits for the first bit of its time slot before it starts transmitting. Data is sent only during the time slots defined by the TSA. Secondly, when working with its own set of signals, the transmitter starts sending when $\overline{\text{SMSYN}x}$ is asserted.

When a BD data is completely written to the transmit FIFO, the L bit is checked and if it is set, the SMC writes the message status bits into the BD and clears the R bit. It then starts transmitting idles. When the end of the current BD is reached and the L bit is not set, only R is cleared. In both cases, an interrupt is issued according to the I bit in the BD. By appropriately setting the I bit in each BD, interrupts can be generated after each buffer, a specific buffer, or each block is sent. The SMC then proceeds to the next BD. If no additional buffers have been presented to the SMC for transmission and the L bit was cleared, an underrun is detected and the SMC begins sending idles.

If the CM bit is set in the TxBD, the R bit is not cleared, so the CP can overwrite the buffer on its next access. For instance, if a single TxBD is initialized with the CM and W bits set, the buffer is sent continuously until R is cleared in the BD.

28.4.3 SMC Transparent Channel Reception Process

When the core enables the SMC receiver in transparent mode, it waits for synchronization before receiving data. Once synchronization is achieved, the receiver transfers the incoming data into memory according to the first RxBD in the table. Synchronization can be achieved in two ways. First, when the receiver is connected to a TDM channel, it can be synchronized to a time slot. Once the frame sync is received, the receiver waits for the first bit of its time slot to occur before reception begins. Data is received only during the time slots defined by the TSA. Secondly, when working with its own set of signals, the receiver starts reception when $\overline{\text{SMSYN}x}$ is asserted.

When the buffer full, the SMC clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the data buffer length, the SMC fetches the next BD; if it is empty, the

SMC continues transferring data to this BD’s buffer. If the CM bit is set in the RxBD, the E bit is not cleared, so the CP can automatically overwrite the buffer on its next access.

28.4.4 Using $\overline{\text{SMSYN}}$ for Synchronization

The $\overline{\text{SMSYN}}$ signal offers a way to externally synchronize the SMC channel. This method differs somewhat from the synchronization options available in the SCCs and should be studied carefully. See Figure 28-11 for an example.

Once SMCMR[REN] is set, the first rising edge of SMCLK that finds $\overline{\text{SMSYN}}$ low causes the SMC receiver to achieve synchronization. Data starts being received or latched on the same rising edge of SMCLK that latched $\overline{\text{SMSYN}}$. This is the first bit of data received. The receiver does not lose synchronization again, regardless of the state of $\overline{\text{SMSYN}}$, until REN is cleared.

Once SMCMR[TEN] is set, the first rising edge of SMCLK that finds $\overline{\text{SMSYN}}$ low synchronizes the SMC transmitter which begins sending ones asynchronously from the falling edge of $\overline{\text{SMSYN}}$. After one character of ones is sent, if the transmit FIFO is loaded (the TxBD is ready with data), data starts being send on the next falling edge of SMCLK after one character of ones is sent. If the transmit FIFO is loaded later, data starts being sent after some multiple number of all-ones characters is sent.

Note that regardless of whether the transmitter or receiver uses $\overline{\text{SMSYN}}$, it must make glitch-free transitions from high-to-low or low-to-high. Glitches on $\overline{\text{SMSYN}}$ can cause errant behavior of the SMC.

The transmitter never loses synchronization again, regardless of the state of $\overline{\text{SMSYN}}$, until the TEN bit is cleared or an ENTER HUNT MODE command is issued.

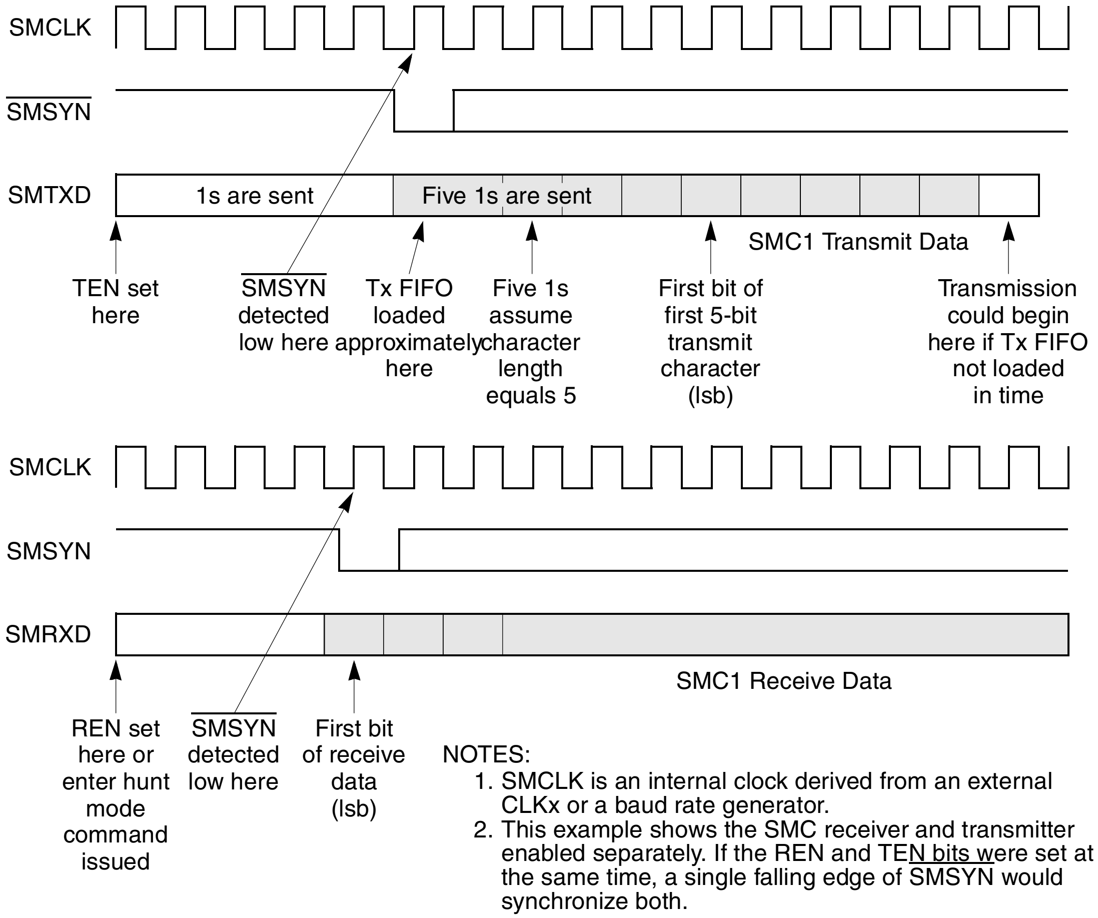

SMCLK
SMSYN
SMTXD
1s are sent
Five 1s are sent
SMC1 Transmit Data
TEN set here
SMSYN detected low here
Tx FIFO loaded approximately here
Five 1s assume character length equals 5
First bit of first 5-bit transmit character (lsb)
Transmission could begin here if Tx FIFO not loaded in time
SMCLK
SMSYN
SMRXD
SMC1 Receive Data
REN set here or enter hunt mode command issued
SMSYN detected low here
First bit of receive data (lsb)
NOTES:
1. SMCLK is an internal clock derived from an external CLKx or a baud rate generator.
2. This example shows the SMC receiver and transmitter enabled separately. If the REN and TEN bits were set at the same time, a single falling edge of SMSYN would synchronize both.

Figure 28-11. Synchronization with SMSYN*x*

If both SMCMR[REN] and SMCMR[TEN] are set, the first falling edge of SMSYN causes both the transmitter and receiver to achieve synchronization. The SMC transmitter can be disabled and reenabled and SMSYN can be used again to resynchronize the transmitter itself. Section 28.2.4, “Disabling SMCs On-the-Fly,” describes how to safely disable and reenable the SMC. Simply clearing and setting TEN may be insufficient. The receiver can also be resynchronized this way.

28.4.5 Using the Time-Slot Assigner (TSA) for Synchronization

The TSA offers an alternative to using SMSYN to internally synchronize the SMC channel. This method is similar, except that the synchronization event is the first time slot for this SMC receiver/transmitter after the frame sync indication rather than the falling edge of SMSYN. Chapter 14, “Serial Interface with Time-Slot Assigner,” describes how to configure time slots. The TSA allows the SMC receiver and transmitter to be enabled simultaneously and synchronized separately; SMSYN does not provide this capability. Figure 28-12 shows synchronization using the TSA.

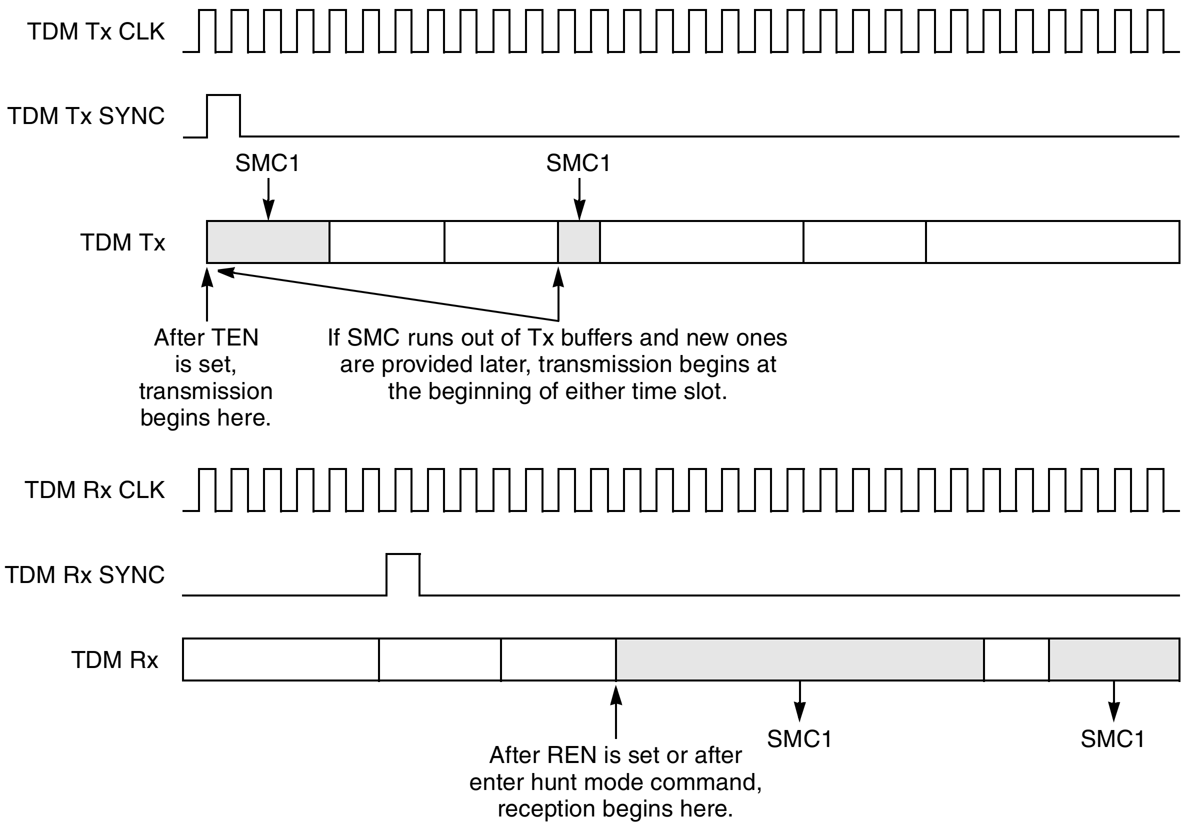

TDM Tx CLK
TDM Tx SYNC
SMC1
SMC1
TDM Tx
After TEN is set, transmission begins here.
If SMC runs out of Tx buffers and new ones are provided later, transmission begins at the beginning of either time slot.
TDM Rx CLK
TDM Rx SYNC
TDM Rx
After REN is set or after enter hunt mode command, reception begins here.
SMC1
SMC1

Figure 28-12. Synchronization with the TSA

Once SMCMR[REN] is set, the first time slot after the frame sync causes the SMC receiver to achieve synchronization. Data is received immediately, but only during defined receive time slots. The receiver continues receiving data during its defined time slots until REN is cleared. If an ENTER HUNT MODE command is issued, the receiver loses synchronization, closes the buffer, and resynchronizes to the first time slot after the frame sync.

Once SMCMR[TEN] is set, the SMC waits for the transmit FIFO to be loaded before trying to achieve synchronization. When the transmit FIFO is loaded, synchronization and transmission begins depending on the following:

- If a buffer is made ready when the SMC2 is enabled, the first byte is placed in time slot 1 if CLSN is 8 and to slot 2 if CLSN is 16.
- If a buffer has its SMC enabled, then the first byte in the next buffer can appear in any time slot associated with this channel.
- If a buffer is ended with the L bit set, then the next buffer can appear in any time slot associated with this channel.

If the SMC runs out of transmit buffers and a new buffer is provided later, idles are sent in the gap between buffers. Data transmission from the later buffer begins at the start of an SMC time slot, but not necessarily the first time slot after the frame sync. So, to maintain a certain bit alignment beginning with the first time slot, make sure that at least one TxBD is always ready and that underruns do not occur. Otherwise, the SMC transmitter should be disabled and reenabled. Section 28.2.4, “Disabling SMCs On-the-Fly,” describes how to safely disable and reenable the SMC. Simply clearing and setting TEN may not be enough.

28.4.6 SMC Transparent Commands

Table 28-10 describes transmit commands issued to the CPCR.

Table 28-10. SMC Transparent Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the SMCM, the channel is in transmit enable mode and polls the first BD. This command disables transmission of frames on the transmit channel. If the transparent controller receives this command while sending a frame, it stops after the contents of the FIFO are sent (up to 2 characters). The TBPTR is not advanced to the next BD, no new BD is accessed, and no new buffers are sent for this channel. The transmitter sends idles until a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Starts or resumes transmission from the current TBPTR in the channel TxBD table. When the channel receives this command, it polls the R bit in this BD. The SMC expects this command after a STOP TRANSMIT is issued. The channel in its mode register is disabled or after a transmitter error occurs.
INIT TX PARAMETERS	Initializes transmit parameters in this serial channel to reset state. Use only if the transmitter is disabled. The INIT TX AND RX PARAMETERS command resets transmit and receive parameters.

Table 28-11 describes receive commands issued to the CPCR.

Table 28-11. SMC Transparent Receive Commands

Command	Description
ENTER HUNT MODE	Forces the SMC to close the current receive BD if it is in use and to use the next BD for subsequent data. If the SMC is not receiving data, the buffer is not closed. Additionally, this command causes the receiver to wait for a resynchronization before reception resumes.
CLOSE RXBD	Forces the SMC to close the current receive BD if it in use and to use the next BD in the list for subsequent received data. If the SMC is not in the process of receiving data, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel to reset state. Use only if the receiver is disabled. The INIT TX AND RX PARAMETERS command resets receive and transmit parameters.

28.4.7 Handling Errors in the SMC Transparent Controller

The SMC uses BDs and the SMCE to report message send and receive errors.

Table 28-12. SMC Transparent Error Conditions

Error	Descriptions
Underrun	The channel stops sending the buffer, closes it, sets UN in the BD, and generates a TXE interrupt if it is enabled. The channel resumes sending after a RESTART TRANSMIT command. Underrun cannot occur between frames.
Overrun	The SMC maintains an internal FIFO for receiving data. If the buffer is in external memory, the CP begins programming the SDMA channel when the first character is received into the FIFO. If a FIFO overrun occurs, the SMC writes the received data character over the previously received character. The previous character and its status bits are lost. Then the channel closes the buffer, sets OV in the BD, and generates the RXB interrupt if it is enabled. Reception continues as normal.

28.4.8 SMC Transparent RxBD

Using BDs, the CP reports information about the received data for each buffer and closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

- An overrun error occurs.
- A full receive buffer is detected.
- The ENTER HUNT MODE command is issued.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	—		**CM**	—							OV	—
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 28-13. SMC Transparent RxBD

Table 28-13 describes SMC transparent RxBD fields.

Table 28-13. SMC Transparent RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer is empty or is receiving data. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (last BD in RxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to. The number of RxBDs is determined only by the W bit and overall space constraints of the dual-port RAM.

Table 28-13. SMC Transparent RxBD Field Descriptions (continued)

Bits	Name	Description
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 SMCE[RXB] is set when the CP completely fills this buffer indicating that the core must process the buffer. The RXB bit can cause an interrupt if it is enabled.
4–5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the buffer to be overwritten when the CP next accesses this BD. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7–13	—	Reserved, should be cleared
14	OV	Overrun. Set when a receiver overrun occurs during reception. The CP writes OV after the received data is placed into the buffer.
15	—	Reserved, should be cleared

Data length and buffer pointer fields are described in Section 19.2, "SCC Buffer Descriptors (BDs)."

28.4.9 SMC Transparent TxBD

Data is sent to the CP for transmission on an SMC channel by arranging it in buffers referenced by the channel TxBD table. The CP uses BDs to confirm transmission or indicate error conditions so the processor knows buffers have been serviced.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	—	**CM**	—							UN	—
Offset + 2	**Data Length**															
Offset + 4	**Tx Data Buffer Pointer**															
Offset + 6																

Table 28-14. SMC Transparent TxBD

Table 28-15 describes SMC transparent TxBD fields.

Table 28-15. SMC Transparent TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. The BD and buffer can be updated. The CP clears R after the buffer is sent or after an error occurs. 1 The user-prepared data buffer is not sent or is being sent. BD fields cannot be updated if R is set.
1	—	Reserved, should be cleared

Table 28-15. SMC Transparent TxBD Field Descriptions (continued)

Bits	Name	Description
2	**W**	Wrap (final BD in table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is programmable and determined by theW bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is serviced; SMCE[TXE] is unaffected. 1 SMCE[TXB] or SMCE[TXE] are set when the buffer is serviced. They can cause interrupts if they are enabled.
4	**L**	Last in message. 0 The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer (if ready) is sent immediately after the last byte of this buffer. 1 The last byte in this buffer is the last byte in the transmitted transparent frame. After this buffer is sent, the transmitter requires synchronization before the next buffer is sent.
5	—	Reserved, should be cleared
6	**CM**	Continuous mode. 0 Normal operation 1 The CP does not clear R after this BD is closed, allowing the buffer to be automatically resent when the CP accesses this BD again. However, the R bit is cleared if an error occurs during transmission, regardless of how CM is set.
7–13	—	Reserved, should be cleared
14	UM	Underrun. Set when the SMC encounters a transmitter underrun condition while sending the buffer.
15	—	Reserved, should be cleared

Data length represents the number of octets the CP should transmit from this buffer. It is never modified by the CP. The data length can be even or odd, but if the number of bits in the transparent character is greater than 8, the data length should be even. For example, to transmit three transparent 8-bit characters, the data length field should be initialized to 3. However, to transmit three transparent 9-bit characters, the data length field should be initialized to 6 because the three 9-bit characters occupy three half words in memory.

The data buffer pointer points to the first byte of the buffer. They can be even or odd, unless character length is greater than 8 bits, in which case the transmit buffer pointer must be even. For instance, the pointer to 8-bit transparent characters can be even or odd, but the pointer to 9-bit transparent characters must be even. The buffer can reside in internal or external memory.

28.4.10 SMC Transparent Event Register (SMCE)/Mask Register (SMCM)

The SMC event register (SMCE) generates interrupts and reports events recognized by the SMC channel. When an event is recognized, the SMC sets the corresponding SMCE bit. Interrupts are masked in the SMCM, which has the same format as the SMCE. SMCE bits are cleared by writing a 1 (writing 0 has no effect). Unmasked bits must be cleared before the CP clears the internal interrupt request. The SMCE and SMCM registers are displayed in Figure 28-14.

	0	1	2	3	4	5	6	7
Field	—			TXE	—	BSY	TXB	RXB
Reset	0							
R/W	R/W							
Addr	0x11A86 (SMCE1), 0x11A96 (SMCE2)/ 0x11A8A (SMCM1), 0x11A9A (SMCM2)							

Figure 28-14. SMC Transparent Event Register (SMCE)/Mask Register (SMCM)

Table 28-16 describes SMCE/SMCM fields.

Table 28-16. SMCE/SMCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared
3	TXE	Tx error. Set when an underrun error occurs on the transmitter channel. This event is not maskable via the TxBD[I] bit.
4	—	Reserved, should be cleared
5	BSY	Busy condition. Set when a character is received and discarded due to a lack of buffers. Reception begins after a new buffer is provided. Executing an ENTER HUNT MODE command makes the receiver wait for resynchronization.
6	TXB	Tx buffer. Set after a buffer is sent. If the L bit of the TxBD is set, TXB is set when the last character starts being sent. A one character-time delay is required to ensure that data is completely sent over the transmit signal. If the L bit of the TxBD is cleared, TXB is set when the last character is written to the transmit FIFO. A two character-time delay is required to ensure that data is completely sent.
7	RXB	Rx buffer. Set when a buffer is received (after the last character is written) on the SMC channel and its associated RxBD is now closed.

28.4.11 SMC Transparent NMSI Programming Example

The following example initializes the SMC1 transparent channel over its own set of signals. The CLK9 signal supplies the transmit and receive clocks; the $\overline{\text{SMSYN}x}$ signal is used for synchronization. (The SMC UART programming example uses a BRG configuration; see Section 28.3.12, “SMC UART Controller Programming Example.”)

1. Configure the port D pins to enable SMTXD1, SMRXD1, and $\overline{\text{SMSYN1}}$. Set PPARD[7,8,9] and PDIRD[9]. Clear PDIRD[7,8] and PSORD[7,8,9].
2. Configure the port C pins to enable CLK9. Set PPARC[23]. Clear PDIRC[23] and PSORC[23].
3. Connect CLK9 to SMC1 using the CPM mux. Clear CMXSMR[SMC1] and program CMXSMR[SMC1CS] to 0b11.
4. In address 0x87FC, assign a pointer to the SMC1 parameter RAM.
5. Write RBASE and TBASE in the SMC parameter RAM to point to the RxBD and TxBD in the dual-port RAM. Assuming one RxBD at the beginning of the dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
6. Write 0x1D01_0000 to CPCR to execute the INIT RX AND TX PARAMETERS command.
7. Write RFCR and TFCR with 0x10 for normal operation.

8. Write MRBLR with the maximum bytes per receive buffer. Assuming 16 bytes MRBLR = 0x0010.
9. Initialize the RxBD assuming the buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
10. Initialize the TxBD assuming the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
11. Write 0xFF to SMCE1 to clear any previous events.
12. Write 0x13 to SMCM1 to enable all possible SMC1 interrupts.
13. Write 0x0000_1000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Write 0xFFFF_FFFF to the SIU interrupt pending register low (SIPNR_L) to clear events.
14. Write 0x3830 to the SMCMR to configure 8-bit characters, unreversed data, and normal operation (not loopback). The transmitter and receiver are not enabled yet.
15. Write 0x3833 to the SMCMR to enable the SMC transmitter and receiver. This additional write ensures that TEN and REN are enabled last.

After 5 bytes are sent, the TxBD is closed; after 16 bytes are received the receive buffer is closed. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

28.5 The SMC in GCI Mode

The SMC can control the C/I and monitor channels of the GCI frame. When using the SCIT configuration of a GCI, one SMC can handle SCIT channel 0 and the other can handle SCIT channel 1. The main features of the SMC in GCI mode are as follows:

- Each SMC channel supports the C/I and monitor channels of the GCI (IOM-2) in ISDN applications.
- Two SMCs support both sets of C/I and monitor channels in SCIT channels 0 and 1.
- Full-duplex operation
- Local loopback and echo capability for testing

To use the SMC GCI channels properly, the TSA must be configured to route the monitor and C/I channels to the preferred SMC. Chapter 14, “Serial Interface with Time-Slot Assigner,” describes how to program this configuration. GCI mode is selected by setting SMCMR[SM] to 0b10. Section 28.2.1, “SMC Mode Registers (SMCMR1/SMCMR2)” describes other protocol-specific SMCMR bits.

28.5.1 SMC GCI Parameter RAM

The GCI parameter RAM differs from that for UART and transparent mode. The CP accesses each SMC’s GCI parameter table using a user-programmed pointer (SMC*x*_BASE) located in the parameter RAM; see Section 13.5.2, “Parameter RAM.” Each SMC GCI parameter RAM table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11 and 12). In GCI mode, parameter RAM contains the BDs instead of pointers to them. Compare Table 28-17 with Table 28-2 on page 28-6 to see the differences. (In GCI mode, the SMC has no extra protocol-specific parameter RAM.)

Table 28-17. SMC GCI Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**M_RxBD**	Half word	Monitor channel RxBD. See Section 28.5.5, “SMC GCI Monitor Channel RxBD.”
0x02	**M_TxBD**	Half word	Monitor channel TxBD. See Section 28.5.6, “SMC GCI Monitor Channel TxBD.”
0x04	**CI_RxBD**	Half word	C/I channel RxBD. See Section 28.5.7, “SMC GCI C/I Channel RxBD.”
0x06	**CI_TxBD**	Half word	C/I channel TxBD. See Section 28.5.8, “SMC GCI C/I Channel TxBD.”
0x08	RSTATE[2]	Word	Rx/Tx internal state
0x0C	M_RxD [2]	Half word	Monitor Rx data
0x0E	M_TxD [2]	Half word	Monitor Tx data
0x10	CI_RxD [2]	Half word	C/I Rx data
0x12	CI_TxD [2]	Half word	C/I Tx data

[1] From the pointer value programmed in SMC*x*_BASE: SMC1_BASE at 0x87FC, SMC2_BASE at 0x88FC.

[2] RSTATE, M_RxD, M_TxD, CI_RxD, and CI_TxD do not need to be accessed by the user in normal operation, and are reserved for RISC use only.

28.5.2 Handling the GCI Monitor Channel

The following sections describe how the GCI monitor channel is handled.

28.5.2.1 SMC GCI Monitor Channel Transmission Process

Monitor channel 0 is used to exchange data with a layer 1 device (reading and writing internal registers and transferring of the S and Q bits). Monitor channel 1 is used for programming and controlling voice/data modules such as CODECs. The core writes the byte into the TxBD. The SMC sends the data on the monitor channel and handles the A and E control bits according to the GCI monitor channel protocol. The TIMEOUT command resolves deadlocks when errors in the A and E bit states occur on the data line.

28.5.2.2 SMC GCI Monitor Channel Reception Process

The SMC receives data and handles the A and E control bits according to the GCI monitor channel protocol. When the CP stores a received data byte in the SMC RxBD, a maskable interrupt is generated. A TRANSMIT ABORT REQUEST command causes the PowerQUICC II to send an abort request on the E bit.

28.5.3 Handling the GCI C/I Channel

The C/I channel is used to control the layer 1 device. The layer 2 device in the TE sends commands and receives indication to or from the upstream layer 1 device through C/I channel 0. In the SCIT configuration, C/I channel 1 is used to convey real-time status information between the layer 2 device and nonlayer 1 peripheral devices (CODECs).

28.5.3.1 SMC GCI C/I Channel Transmission Process

The core writes the data byte into the C/I TxBD and the SMC transmits the data continuously on the C/I channel to the physical layer device.

28.5.3.2 SMC GCI C/I Channel Reception Process

The SMC receiver continuously monitors the C/I channel. When it recognizes a change in the data and this value is received in two successive frames, it is interpreted as valid data. This is called the double last-look method. The CP stores the received data byte in the C/I RxBD and a maskable interrupt is generated. If the SMC is configured to support SCIT channel 1, the double last-look method is not used.

28.5.4 SMC GCI Commands

The commands in Table 28-18 are issued to the CPCR.

Table 28-18. SMC GCI Commands

Command	Description
INIT TX AND RX PARAMETERS	Initializes transmit and receive parameters in the parameter RAM to their reset state. It is especially useful when switching protocols on a given serial channel.
TRANSMIT ABORT REQUEST	This receiver command can be issued when the PowerQUICC II implements the monitor channel protocol. When it is issued, the PowerQUICC II sends an abort request on the A bit.
TIMEOUT	This transmitter command can be issued when the PowerQUICC II implements the monitor channel protocol. It is usually issued because the device is not responding or A bit errors are detected. The PowerQUICC II sends an abort request on the E bit at the time this command is issued.

28.5.5 SMC GCI Monitor Channel RxBD

This BD, seen in Figure 28-15, is used by the CP to report information about the monitor channel receive byte.

	0	1	2	3	4 7	8 15
Offset + 0	E	I	ER	MS	—	DATA

Figure 28-15. SMC Monitor Channel RxBD

Table 28-19 describes SMC monitor channel RxBD fields.

Table 28-19. SMC Monitor Channel RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The CP clears E when the byte associated with this BD is available to the core. 1 The core sets E when the byte associated with this BD has been read.
1	L	Last (EOM). Valid only for monitor channel protocol and is set when the EOM indication is received on the E bit. Note that when this bit is set, the data byte is invalid.
2	**ER**	Error condition. Valid only for monitor channel protocol. Set when an error occurs on the monitor channel protocol. A new byte is sent before the SMC acknowledges the previous byte.
3	**MS**	Data mismatch. Valid only for monitor channel protocol. Set when two different consecutive bytes are received; cleared when the last two consecutive bytes match. The SMC waits for the reception of two identical consecutive bytes before writing new data to the RxBD.
4–7	—	Reserved, should be cleared
8–15	DATA	Data field. Contains the monitor channel data byte that the SMC received.

28.5.6 SMC GCI Monitor Channel TxBD

The CP uses this BD, shown in Figure 28-16, to report about the monitor channel transmit byte.

	0	1	2	3	7	8	15
Offset + 0	**R**	L	**AR**	—		DATA	

Figure 28-16. SMC Monitor Channel TxBD

Table 28-20 describes SMC monitor channel TxBD fields.

Table 28-20. SMC Monitor Channel TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 Cleared by the CP after transmission. The TxBD is now available to the core. 1 Set by the core when the data byte associated with this BD is ready for transmission.
1	L	Last (EOM). Valid only for monitor channel protocol. When L = 1, the SMC first transmits the buffer data and then transmits the EOM indication on the E bit.
2	**AR**	Abort request. Valid only for monitor channel protocol. Set by the SMC when an abort request is received on the A bit. The transmitter sends the EOM on the E bit after receiving an abort request.
3–7	—	Reserved, should be cleared
8–15	DATA	Data field. Contains the data to be sent by the SMC on the monitor channel.

28.5.7 SMC GCI C/I Channel RxBD

The CP uses this BD, seen in Figure 28-17, to report information about the C/I channel receive byte.

<table>
<tr><th></th><th>0</th><th>1</th><th></th><th></th><th></th><th></th><th></th><th>7</th><th>8</th><th></th><th></th><th></th><th></th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0</td><td>E</td><td colspan="7">—</td><td colspan="6">C/I DATA</td><td colspan="2">—</td></tr>
</table>

Figure 28-17. SMC C/I Channel RxBD

Table 28-21 describes SMC C/I channel RxBD fields.

Table 28-21. SMC C/I Channel RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 Cleared by the CP to indicate that the byte associated with this BD is available to the core. 1 The core sets E to indicate that the byte associated with this BD has been read. Note that additional data received is discarded until E bit is set.
1–7	—	Reserved, should be cleared
8–13	C/I DATA	Command/indication data bits. For C/I channel 0, bits 10–13 contain the 4-bit data field and bits 8–9 are always written with zeros. For C/I channel 1, bits 8–13 contain the 6-bit data field.
14–15	—	Reserved, should be cleared

28.5.8 SMC GCI C/I Channel TxBD

The CP uses this BD, as seen in Figure 28-18, to report about the C/I channel transmit byte.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0</td><td>R</td><td colspan="7">—</td><td colspan="6">C/I DATA</td><td colspan="2">—</td></tr>
</table>

Figure 28-18. SMC C/I Channel TxBD

Table 28-22 describes SMC C/I channel TxBD fields.

Table 28-22. SMC C/I Channel TxBD Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 Cleared by the CP after transmission to indicate that the BD is available to the core. 1 Set by the core when data associated with this BD is ready for transmission.
1–7	—	Reserved, should be cleared
8–13	C/I DATA	Command/indication data bits. For C/I channel 0, bits 10–13 hold the 4-bit data field (bits 8 and 9 are always written with zeros). For C/I channel 1, bits 8–13 contain the 6-bit data field.
14–15	—	Reserved, should be cleared.

28.5.9 SMC GCI Event Register (SMCE)/Mask Register (SMCM)

The SMCE generates interrupts and report events recognized by the SMC channel. When an event is recognized, the SMC sets its corresponding SMCE bit. SMCE bits are cleared by writing ones; writing zeros has no effect. SMCM has the same bit format as SMCE. Setting an SMCM bit enables, and clearing an SMCM bit disables, the corresponding interrupt. Unmasked bits must be cleared before the CP clears the internal interrupt request to the SIU interrupt controller. Figure 28-19 displays the SMCE/SMCM registers.

	0	1	2	3	4	5	6	7
Field	—				CTXB	CRXB	MTXB	MRXB
Reset	0000_0000							
R/W	R/W							
Addr	0x11A86 (SMCE1), 0x11A96 (SMCE2)/ 0x11A8A (SMCM1), 0x11A9A (SMCM2)							

Figure 28-19. SMC GCI Event Register (SMCE)/Mask Register (SMCM)

Table 28-23 describes SMCE/SMCM fields.

Table 28-23. SMCE/SMCM Field Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	CTXB	C/I channel buffer transmitted. Set when the C/I transmit buffer is now empty.
5	CRXB	C/I channel buffer received. Set when the C/I receive buffer is full.
6	MTXB	Monitor channel buffer transmitted. Set when the monitor transmit buffer is now empty.
7	MRXB	Monitor channel buffer received. Set when the monitor receive buffer is full.

Chapter 29
Fast Communications Controllers (FCCs)

The MPC8272's three fast communications controllers (FCCs) are serial communications controllers (SCCs) optimized for synchronous high-rate protocols. FCC key features include the following:

- Supports HDLC/SDLC and totally transparent protocols
- FCC clocks can be derived from a baud-rate generator or an external signal.
- Supports $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem control signals
- Use of bursts to improve bus usage
- Multibuffer data structure for receive and transmit, external buffer descriptors (BDs) anywhere in system memory
- 192-byte FIFO buffers
- Full-duplex operation
- Fully transparent option for one half of an FCC (receiver/transmitter) while HDLC/SDLC protocol executes on the other half (transmitter/receiver)
- Echo and local loopback modes for testing
- 10/100 Mbps Ethernet through RMII interface
- ATM internal rate mode for 31 PHYs
- ATM 31 PHY addresses for both FCC1 and FCC2
- Assuming a 100-MHz CPM clock, the FCCs support the following:
  - Full 10/100-Mbps Ethernet/IEEE 802.3x through an MII
  - Full 155-Mbps ATM segmentation and reassembly (SAR) through UTOPIA
  - 45-Mbps (DS-3/E3 rates) HDLC and/or transparent data rates supported on each FCC

FCCs differ from SCCs as follows:

- No DPLL support
- No BISYNC, UART, or AppleTalk/LocalTalk support
- No HDLC bus
- Ethernet support only through an MII/RMII

29.1 Overview

MPC8272 FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and interface with a wide variety of standard WANs, LANs, and proprietary networks. FCCs have many physical interface options such as interfacing to TDM buses, ISDN buses, standard modem interfaces, fast Ethernet interface (MII), and ATM interfaces (UTOPIA); see Chapter 14, "Serial Interface with Time-Slot Assigner," Chapter 32, "Fast Ethernet

Controller,” and Chapter 30, “ATM Controller and AAL0, AAL1, and AAL5 Protocols.” The FCCs are independent from the physical interface, but FCC logic formats and manipulates data from the physical interface. That is why the interfaces are described separately.

The FCC is described in terms of the protocol that it is chosen to run. When an FCC is programmed to a certain protocol, it implements a certain level of functionality associated with that protocol. For most protocols, this corresponds to portions of the link layer (layer 2 of the seven-layer OSI model). Many functions of the FCC are common to all of the protocols. These functions are described in the FCC description. Following that, the implementation details that differentiate protocols from one another are discussed, beginning with the transparent protocol. Thus, the reader should read from this point to the transparent protocol and then skip to the appropriate protocol. Since FCCs use similar data structures across all protocols, the reader's learning time decreases dramatically after understanding the first protocol.

Each FCC supports a number of protocols—Ethernet, HDLC/SDLC, ATM, and totally transparent operation. Although the selected protocol usually applies to both the FCC transmitter and receiver, half of one FCC can run transparent operation while the other runs the HDLC/SDLC protocol. The internal clocks (RCLK, TCLK) for each FCC can be programmed with either an external or internal source. The internal clocks originate from one of the baud-rate generators or one of the external clock signals. The limitation of the internal clocks frequency depends on the protocol being used, see Table 29-1. See Chapter 14, “Serial Interface with Time-Slot Assigner.” However, the FCC’s ability to support a sustained bit stream depends on the protocol as well as on other factors.

Each FCC can be connected to its own set of pins on the MPC8272. This configuration, the non-multiplexed serial interface, or NMSI, is described in Chapter 14, “Serial Interface with Time-Slot Assigner.” In this configuration, each FCC can support the standard modem interface signals ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$) through the appropriate port pins and the interrupt controller. Additional handshake signals can be supported with additional parallel I/O lines. The FCC block diagram is shown in Figure 29-1.

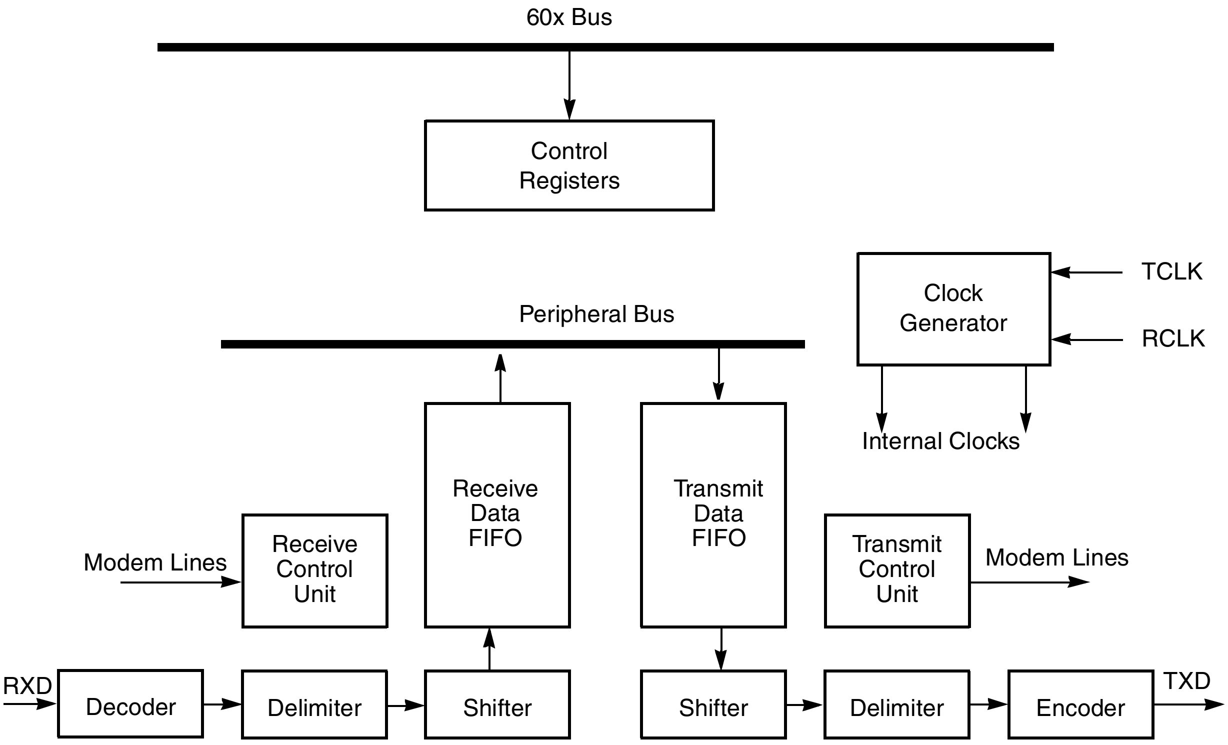


Figure 29-1. FCC Block Diagram

Table 29-1. Internal Clocks to CPM Clock Frequency Ratio

Mode	Internal Clock: CPM Clock Frequency Ratio
HDLC 1 bit	1:4
Transparent 1 bit	1:4
HDLC nibble	1:6
Fast Ethernet	1:3 (1:3.5 preferred)
ATM	1:3 (1:3.5 preferred)

29.2 Mode Registers

29.2.1 General FCC Mode Registers (GFMR*x*)

Each FCC contains a general FCC mode register (GFMR*x*) that defines common FCC options and selects the protocol to be run. The GFMR*x* are read/write registers cleared at reset. Figure 29-2 shows the GFMR format.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DIAG		TCI	TRX	TTX	CDP	CTSP	CDS	CTSS	—						
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11300 (GFMR1), 0x11320(GFMR2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SYNL		RTSM	RENC		REVD	TENC		TCRC		ENR	ENT	MODE			
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11302 (GFMR1), 0x11322 (GFMR2)															

Figure 29-2. General FCC Mode Register (GFMR)

Table 29-2 describes GFMR fields.

Table 29-2. GFMR Register Field Descriptions

Bits	Name	Description
0–1	DIAG	Diagnostic mode. 00 Normal operation—Receive data enters through RXD, and transmit data is shifted out through TXD. The FCC uses the modem signals ($\overline{\text{CD}}$ and $\overline{\text{CTS}}$) to automatically enable and disable transmission and reception. Timings are shown in Section 29.11, "FCC Timing Control." 01 Local loopback mode—Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. RXD is ignored. Data can be programmed to appear on TXD, or TXD can remain high by programming the appropriate parallel port register. $\overline{\text{RTS}}$ can be disabled in the appropriate parallel I/O register. The transmitter and receiver must use the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{\text{RTS}}$ 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 FCC by configuring the associated $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins as general-purpose I/O.; see Chapter 37, "Parallel I/O Ports." 10 Automatic echo mode—The channel automatically retransmits received data, using the receive clock provided. The receiver operates normally and receives data if $\overline{\text{CD}}$ is asserted. The transmitter simply transmits received data. In this mode, $\overline{\text{CTS}}$ is ignored. The echo function can also be accomplished in software by receiving buffers from an FCC, linking them to TxBDs, and transmitting them back out of that FCC. 11 Loopback and echo mode—Loopback and echo operation occur simultaneously. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are ignored. Refer to the loopback bit description for clocking requirements. For TDM operation, the diagnostic mode is selected by SI*x*MR[SDM*x*]; see Section 14.5.2, "SI2 Mode Register (SI2MR)."
2	TCI	Transmit clock invert 0 Normal operation 1 The FCC inverts the internal transmit clock. The edge on which the FCC outputs the data depends on the protocol: • In HDLC and transparent mode, when TCI=0, data is sent on the falling edge; when TCI=1, on the rising edge. • In Ethernet mode, when TCI=0, data is sent on the rising edge; when TCI=1, on the falling edge.

Table 29-2. GFMR Register Field Descriptions (continued)

Bits	Name	Description
3	TRX	Transparent receiver. The MPC8272 FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications such as an FCC transmitter configured to HDLC and a receiver configured to totally transparent operation. To do this, program MODE = HDLC, TTX = 0, and TRX = 1. 0 Normal operation 1 The receiver operates in totally transparent mode, regardless of the protocol selected for the transmitter in the MODE bits. **Note:** Full-duplex, totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its transmitter and transparent operation on its receiver causes erratic behavior. In other words, if the MODE = Ethernet or ATM, TTX must equal TRX.
4	TTX	Transparent transmitter. The MPC8272 FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications, such as configuring an FCC receiver to HDLC and a transmitter to totally transparent operation. To do this, program MODE = HDLC, TTX = 1, and TRX = 0. 0 Normal operation 1 The transmitter operates in totally transparent mode, regardless of the receiver protocol selected in the MODE bits. **Note:** Full-duplex totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its receiver and transparent operation on its transmitter causes erratic behavior. In other words, if GFMR[MODE] selects Ethernet or ATM, TTX must equal TRX.
5	CDP	$\overline{\text{CD}}$ pulse (transparent mode only) 0 Normal operation (envelope mode). $\overline{\text{CD}}$ should envelope the frame; to negate $\overline{\text{CD}}$ while receiving causes a $\overline{\text{CD}}$ lost error. 1 Pulse mode. Once $\overline{\text{CD}}$ is asserted (high to low transition), synchronization has been achieved, and further transitions of $\overline{\text{CD}}$ do not affect reception. **Note:** CDP must be set if this FCC is used with the TSA in transparent mode.
6	CTSP	$\overline{\text{CTS}}$ pulse 0 Normal operation (envelope mode). $\overline{\text{CTS}}$ should envelope the frame; to negate $\overline{\text{CTS}}$ while transmitting causes a $\overline{\text{CTS}}$ lost error. See Section 29.11, "FCC Timing Control." 1 Pulse mode. $\overline{\text{CTS}}$ is asserted when synchronization is achieved; further transitions of $\overline{\text{CTS}}$ do not affect transmission. When running HDLC, the FCC samples $\overline{\text{CTS}}$ only once before sending the first frame after the transmitter is enabled (ENT = 1).
7	CDS	$\overline{\text{CD}}$ sampling 0 The $\overline{\text{CD}}$ input is assumed to be asynchronous with the data. The FCC synchronizes it internally before data is received. (This mode is not allowed in transparent mode when SYNL = 0b00.) 1 The $\overline{\text{CD}}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{\text{CD}}$ must transition while the receive clock is in the low state. When $\overline{\text{CD}}$ goes low, data is received. This is useful when connecting MPC8272s in transparent mode since it allows the $\overline{\text{RTS}}$ signal of one MPC8272 to be connected directly to the $\overline{\text{CD}}$ signal of another MPC8272.

Table 29-2. GFMR Register Field Descriptions (continued)

Bits	Name	Description
8	CTSS	$\overline{\text{CTS}}$ sampling 0 The $\overline{\text{CTS}}$ input is assumed to be asynchronous with the data. When it is internally synchronized by the FCC, data is sent after a delay of no more than two serial clocks. 1 The $\overline{\text{CTS}}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{\text{CTS}}$ must transition while the transmit clock is in the low state. As soon as $\overline{\text{CTS}}$ is low, data transmission begins. This mode is useful when connecting MPC8272 in transparent mode because it allows the $\overline{\text{RTS}}$ signal of one MPC8272 to be connected directly to the $\overline{\text{CTS}}$ signal of another MPC8272.
9–15	—	Reserved, should be 0.
16–17	SYNL	Sync length (transparent mode only). Determines the operation of an FCC receiver configured for totally transparent operation only. See Section 34.3.1, "In-Line Synchronization Pattern." 00 The sync pattern in the FDSR is not used. An external sync signal is used instead ($\overline{\text{CD}}$ signal asserted: high to low transition). 01 Automatic sync (assumes always synchronized, ignores $\overline{\text{CD}}$ signal). 10 8-bit sync. The receiver synchronizes on an 8-bit sync pattern stored in the FDSR. Negation of $\overline{\text{CD}}$ causes CD lost error. 11 16-bit sync. The receiver synchronizes on a 16-bit sync pattern stored in the FDSR. Negation of $\overline{\text{CD}}$ causes CD lost error. **Note:** If SYNL = 1x, CDP should be cleared (not in $\overline{\text{CD}}$ pulse mode).
18	RTSM	$\overline{\text{RTS}}$ mode 0 Send idles between frames as defined by the protocol. $\overline{\text{RTS}}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{\text{RTS}}$ is asserted whenever the FCC is enabled.
19–20	RENC	Receiver decoding method. The user should set RENC = TENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved
21	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 The totally transparent channels on this FCC (either the receiver, transmitter, or both, as defined by TTX and TRX) reverse bit order, transmitting the MSB of each octet first.
22–23	TENC	Transmitter encoding method. The user should set TENC = RENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved
24–25	TCRC	Transparent CRC (totally transparent channel only). Selects the type of frame checking provided on the transparent channels of the FCC (either the receiver, transmitter, or both, as defined by TTX and TRX). This configuration selects a frame check type; the decision to send the frame check is made in the TxBD. Thus, it is not required to send a frame check in transparent mode. If a frame check is not used, the user can ignore any frame check errors generated on the receiver. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1) 01 Reserved 10 32-bit CCITT CRC (Ethernet and HDLC) (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1) 11 Reserved

Table 29-2. GFMR Register Field Descriptions (continued)

Bits	Name	Description
26	ENR	Enable receive. Enables the receiver hardware state machine for this FCC. 0 The receiver is disabled and any data in the receive FIFO buffer is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR may be set or cleared regardless of whether serial clocks are present. Describes how to disable and reenable an FCC. Note that the FCC provides other tools for controlling reception—the ENTER HUNT MODE command, CLOSE RXBD command, and RxBD[E].
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this FCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the transmitter aborts the current character and TXD returns to idle state. Data in the transmit shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. See Section 29.12, "Disabling the FCCs On-the-Fly," for a description of the proper methods to disable and reenable an FCC. Note that the FCC provides other tools for controlling transmission besides the ENT bit—the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, $\overline{\text{CTS}}$ flow control, and TxBD[R].
28–31	MODE	Channel protocol mode 0000 HDLC 0001 Reserved for RAM microcode 0010 Reserved 0011 Reserved for RAM microcode 0100 Reserved 0101 Reserved for RAM microcode 0110 Reserved 0111 Reserved for RAM microcode 1000 Reserved 1001 Reserved for RAM microcode 1010 ATM 1011 Reserved for RAM microcode 1100 Ethernet 11xx Reserved

NOTE

In addition to selecting the correct mode of operation in GFMRx[MODE], the user must issue the appropriate CP command and choose the correct protocol in CPCR (refer to Section 13.4.1, "CP Command Register (CPCR)").

29.2.2 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3	7
Field	TIREM	LPB	CLK	—	
Reset	0000_0000				
R/W	R/W				
Addr	0x11390 (GFEMR1), 0x113B0(GFEMR2)				

Figure 29-3. General FCC Expansion Mode Register (GFEMR)

Table 29-3 describes GFEMR*x* fields.

Table 29-3. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0–3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0–31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI and FIRSR_LO. Underrun status for PHYs[0–31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII Loopback diagnostic mode (Ethernet mode) 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50-MHz input clock from external oscillator (Ethernet mode) 0 50 MHz (for fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared

29.3 FCC Protocol-Specific Mode Registers (FPSMR*x*)

The functionality of the FCC varies according to the protocol selected by GFMR[MODE]. Each FCC has an additional 32-bit, memory-mapped, read/write protocol-specific mode register (FPSMR) that configures them specifically for a chosen mode. The section for each specific protocol describes the FPSMR bits.

29.4 FCC Data Synchronization Registers (FDSR*x*)

Each FCC has a 16-bit, memory-mapped, read/write data synchronization register (FDSR) that specifies the pattern used in the frame synchronization procedure of the synchronous protocols. In the totally transparent protocol, the FDSR should be programmed with the preferred SYNC pattern. For the Ethernet protocol, it should be programmed with 0xD555. For the ATM protocol, FSDRx is used to generate 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.

At reset, FDSRx defaults to 0x7E7E (two HDLC flags), so it does not need to be written for HDLC mode. The FDSR contents are always sent lsb first.

	0							7	8							15
Field	SYN2								SYN1							
Reset	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0
R/W	R/W															
Addr	0x1130C (FDSR1), 0x1132C (FDSR2)															

Figure 29-4. FCC Data Synchronization Register (FDSR)

29.5 FCC Transmit-on-Demand Registers (FTODR*x*)

If no frame is being sent by the FCC, the CP periodically polls the R bit of the next TxBD to see if the user has requested a new frame/buffer to be sent. Polling occurs every 256 serial transmit clocks. The polling algorithm depends on FCC configuration, as shown in the following equations:

$$\text{Fast Ethernet:} \qquad 256 \text{ clocks} / 25 \text{ MHz} = 10\ \mu\text{s}$$

$$\text{10BaseT:} \qquad 256 \text{ clocks} / 2.5 \text{ MHz} = 100\ \mu\text{s}$$

The user, however, can request that the CP begin processing the new frame/buffer without waiting the normal polling time. For immediate processing, after setting TxBD[R], set the transmit-on-demand (TOD) bit in the transmit-on-demand register (FTODR) twice to activate. If TOD is set only once, the new frame/buffer will not be transmitted until the next periodic polling request.

This feature, which decreases the transmission latency of the transmit buffer/frame, is particularly useful in LAN-type protocols where maximum interframe GAP times are limited by the protocol specification. Since the transmit-on-demand feature gives a high priority to the specified TxBD, it can conceivably affect the servicing of the other FCC FIFO buffers. Therefore, it is recommended that the transmit-on-demand feature be used only for a high-priority TxBD and when transmission on this FCC has not occurred for a given time period, which is protocol-dependent.

If a new TxBD is added to the BD table while preceding TxBDs have not completed transmission, the new TxBD is processed immediately after the older TxBDs are sent.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TOD	—														
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11308 (FTODR1), 0x11328 (FTODR2)															

Figure 29-5. FCC Transmit-on-Demand Register (FTODR)

Fields in the FTODR are described in Table 29-4

Table 29-4. FTODR Field Descriptions

Field	Name	Description
0	TOD	Transmit on demand 0 Normal polling 1 The CP gives high priority to the current TxBD and begins sending the frame does without waiting for the normal polling time to check TxBD[R]. TOD is cleared automatically.
1–15	—	Reserved, should be cleared

29.6 FCC Buffer Descriptors

Data associated with each FCC is stored in buffers. Each buffer is referenced by a buffer descriptor (BD). All of the transmit BDs for an FCC are grouped into a TxBD circular table with a programmable length. Likewise, receive BDs form an RxBD table. The user can program the start address of the BD tables anywhere in system memory. See Figure 29-6.

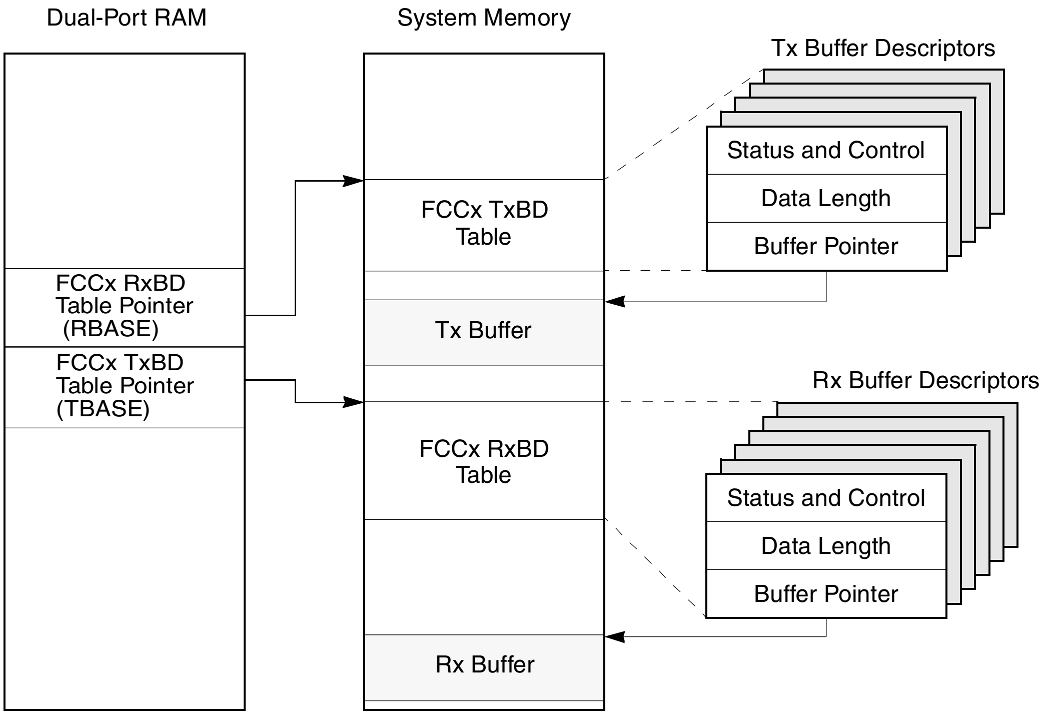


Figure 29-6. FCC Memory Structure

The format of transmit and receive BDs, shown in Figure 29-7, is the same for every FCC mode of operation except ATM mode; see Section 30.10.5, “ATM Controller Buffer Descriptors (BDs).” The first 16 bits in each BD contain status and control information, which differs for each protocol. The second 16 bits indicate the data buffer length in bytes (the wrap bit is the BD table length indicator). The remaining 32-bits contain the 32-bit address pointer to the actual buffer in memory.

	0 — 15
Offset + 0	Status and Control
Offset + 2	Data Length
Offset + 4	High-Order Data Buffer Pointer
Offset + 6	Low-Order Data Buffer Pointer

Figure 29-7. Buffer Descriptor Format

For frame-based protocols, a message can reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (64K – 1) bytes. The CP does not assume that all buffers of a single frame are currently linked to the BD table. It does assume, however, that unlinked buffers are provided by the core soon enough to be sent or received. Failure to do so causes an error condition being reported by the CP. An underrun error is reported in the case of transmit; a busy error is reported in the case of receive. Because BDs are prefetched, the receive BD table must always contain at least one empty BD to avoid a busy error; therefore, RxBD tables must always have at least two BDs.

The BDs and data buffers can be anywhere in the system memory.

The CP processes the TxBDs in a straightforward fashion. Once the transmit side of an FCC is enabled, it starts with the first BD in that FCC’s TxBD table. When the CP detects that TxBD[R] is set, it begins processing the buffer. The CP detects that the BD is ready either by polling the R bit periodically or by the user writing to the FTODR. When the data from the BD has been placed in the transmit FIFO buffer, the CP moves on to the next BD, again waiting for the R bit to be set. Thus, the CP does no look-ahead BD processing, nor does it skip over BDs that are not ready. When the CP sees the wrap (W) bit set in a BD, it goes back to the beginning of the BD table after processing of the BD is complete.

After using a BD, the CP normally clears R (not-ready); thus, the CP does not use a BD again until the BD has been prepared by the core. Some protocols support continuous mode, which allows repeated transmission and for which the R bit remains set (always ready).

The CP uses RxBDs in a similar fashion. Once the receive side of an FCC is enabled, it starts with the first BD in the FCC’s RxBD table. Once data arrives from the serial line into the FCC, the CP performs the required protocol processing on the data and moves the resultant data to the buffer pointed to by the first BD. Use of a BD is complete when no room is left in the buffer or when certain events occur, such as the detection of an error or end-of-frame. Regardless of the reason, the buffer is then said to be closed and additional data is stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it checks the E bit of that BD. This check is made on a prefetched copy of the current BD. If the current BD is not empty, it reports a busy error. However, it does not move from the current BD until it is empty. Because there is a periodic prefetch of the RxBD, the busy error may recur if the BD is not prepared soon enough.

When the CP sees the W bit set in a BD, it returns to the beginning of the BD table after processing of the BD is complete. After using a BD, the CP clears the E bit (not empty) and does not use a BD again until the BD has been processed by the core. However, in continuous mode, available to some protocols, the E bit remains set (always empty).

29.7 FCC Parameter RAM

Each FCC parameter RAM area begins at the same offset from each FCC base area. The protocol-specific portions of the FCC parameter RAM are discussed in the specific protocol descriptions. Table 29-5 shows portions common to all FCC protocols.

Some parameter RAM values must be initialized before the FCC is enabled; other values are initialized/written by the CP. Once initialized, most parameter RAM values do not need to be accessed by user software because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.
- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 29.12, “Disabling the FCCs On-the-Fly.”

Some parameters in Table 29-5 are not described and are listed only to provide information for experienced users and for debugging. The user need not access these parameters in normal operation.

Table 29-5. FCC Parameter RAM Common to All Protocols Except ATM

Offset[1]	Name	Width	Description
0x00	**RIPTR**	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. 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 FCC receiver writes to a receive buffer before moving to the next buffer. The receiver can write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR value. Therefore, user-supplied buffers should be at least as large as the MRBLR. Note that FCC transmit buffers can have varying lengths by programming TxBD[Data Length], as needed, and are not affected by the value in MRBLR. MRBLR is not intended to be changed dynamically while an FCC is operating. Change MRBLR only when the FCC receiver is disabled.
0x08	**RSTATE**	Word	Receive internal state. The high byte, RSTATE[0–7], contains the function code register; see Section 29.7.1, “FCC Function Code Registers (FCRx).” RSTATE[8–31] is used by the CP and must be cleared initially.

Table 29-5. FCC Parameter RAM Common to All Protocols Except ATM (continued)

Offset[1]	Name	Width	Description
0x0C	**RBASE**	Word	RxBD base address (must be divisible by eight). Defines the starting location in the memory map for the FCC RxBDs. This provides great flexibility in how FCC RxBDs are partitioned. By selecting RBASE entries for all FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the receive side of every FCC. The user must initialize RBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x10	RBDSTAT	Hword	RxBD status and control. Reserved for CP use only.
0x12	RBDLEN	Hword	RxBD data length. A down-count value initialized by the CP with MRBLR and decremented with every byte written by the SDMA channels.
0x14	RDPTR	Word	RxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x18	**TSTATE**	Word	Tx internal state. The high byte, TSTATE[0–7], contains the function code register; see Section 29.7.1, “FCC Function Code Registers (FCRx).” TSTATE[8–31] is used by the CP 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 FCC TxBDs. This provides great flexibility in how FCC TxBDs are partitioned. By selecting TBASE entries for all FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the transmit side of every FCC. The user must initialize TBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x20	TBDSTAT	Hword	TxBD status and control. Reserved for CP use only.
0x22	TBDLEN	Hword	TxBD data length. A down-count value initialized with the TxBD data length and decremented with every byte read by the SDMA channels.
0x24	TDPTR	Word	TxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x28	RBPTR	Word	RxBD pointer. Points to the next BD that the receiver transfers data to when it is in idle state or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP sets RBPTR = RBASE. Although the user need never write to RBPTR in most applications, the user can modify it when the receiver is disabled or when no receive buffer is in use.
0x2C	TBPTR	Word	TxBD pointer. Points either to the next BD that the transmitter transfers data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP sets TBPTR = TBASE. Although the user need never write to TBPTR in most applications, the user can modify it when the transmitter is disabled or when no transmit buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes transmission).
0x30	RCRC	Word	Temporary receive CRC
0x34	—	Word	Reserved
0x38	TCRC	Word	Temporary transmit CRC
0x3C	—	Word	First word of protocol-specific area

[1] Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 13.5.2, “Parameter RAM.”

29.7.1 FCC Function Code Registers (FCR*x*)

The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 29-8 shows the format of the transmit and receive function code registers, which reside at TSTATE[0–7] and RSTATE[0–7] in the FCC parameter RAM (see Table 29-5).

	0	1	2	3	4	5	6	7
Field	—	FCCP	GBL	BO		TC2	DTB	BDB

Figure 29-8. Function Code Register (FCR*x*)

FCR*x* fields are described in Table 29-6.

Table 29-6. FCR*x* Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	FCCP	FCC priority. Used in conjunction with PPC_ACR[PRKM] (see Section 4.3.2.2) and LCL_ACR[PRKM] (see Section 4.3.2.4) for a low request level. 0 Disables CPM low request level to refer to FCCs 1 Enables CPM low request level to refer to FCCs.
2	**GBL**	Global. Indicates whether the memory operation should be snooped. 0 Snooping disabled 1 Snooping enabled
3–4	**BO**	Byte ordering. Used to select the byte ordering of the buffer. If BO is modified on-the-fly, it takes effect at the start of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 01 Munged little-endian byte ordering. As data is sent onto the serial line from the data buffer, the LSB of the buffer double-word contains data to be sent earlier than the MSB of the same buffer double-word. 10 Freescale byte ordering (normal operation). It is also called 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	**TC2**	Transfer code. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Indicates on what bus the data is located. 0 On the 60x bus 1 Reserved
7	**BDB**	Indicates on what bus the BDs are located. 0 On the 60x bus 1 Reserved

29.8 Interrupts from the FCCs

Interrupt handling for each of the FCC channels is configured on a global (per channel) basis in the interrupt pending register (SIPNR_L) and interrupt mask register (SIMR_L). One bit in each register is used to either mask, enable, or report an interrupt in an FCC channel. The interrupt priority between the FCCs is programmable in the CPM interrupt priority register (SCPRR_H). The interrupt vector register

(SIVEC) indicates which pending channel has highest priority. Registers within the FCCs manage interrupt handling for FCC-specific events.

Events that can cause the FCC to interrupt the processor vary slightly among protocols and are described with each protocol. These events are handled independently for each channel by the FCC event and mask registers (FCCE and FCCM).

29.8.1 FCC Event Registers (FCCE*x*)

Each FCC has an FCC event register (FCCE) used to report events. On recognition of an event, the FCC sets its corresponding FCCE bit regardless of the corresponding mask bit. To the user it appears as a memory-mapped register that can be read at any time. Bits are cleared by writing ones; writing zeros has no effect on bit values. FCCE is cleared at reset. Fields of this register are protocol-dependent and are described in the respective protocol sections.

29.8.2 FCC Mask Registers (FCCM*x*)

Each FCC has a read/write FCC mask register (FCCM) used to enable or disable CP interrupts to the core for events reported in an event register (FCCE). Bit positions in FCCM are identical to those in FCCE. Note that an interrupt is generated only if the FCC interrupts are also enabled in the SIU; see Section 4.3.1.5, “SIU Interrupt Mask Registers (SIMR_H and SIMR_L).”

If an FCCM bit is zero, the CP does not proceed with its usual interrupt handling whenever that event occurs. Any time a bit in the FCCM register is set, a 1 in the corresponding bit in the FCCE register sets the FCC event bit in the interrupt pending register; see Section 4.3.1.4, “SIU Interrupt Pending Registers (SIPNR_H and SIPNR_L).”

29.8.3 FCC Status Registers (FCCS*x*)

Each FCC has an 8-bit, read/write FCC status register (FCCS) that lets the user monitor real-time status conditions (flags, idle) on the RXD line. It does not show the status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$; their real-time status is available in the appropriate parallel I/O port (see Chapter 37, “Parallel I/O Ports”).

29.9 FCC Initialization

The FCCs require a number of registers and parameters to be configured after a power-on reset. The following outline gives the proper sequence for initializing the FCCs, regardless of the protocol used.

1. Write the parallel I/O ports to configure and connect the I/O pins to the FCCs.
2. Write the appropriate port registers to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to be parallel I/O with interrupt capability or to connect directly to the FCC (if modem support is needed).
3. If the TSA is used, the SI must be configured. If the FCC is used in the NMSI mode, the CPM multiplexing logic (CMX) must still be initialized.
4. Write the GFMR, but do not write the ENT or ENR bits yet.
5. Write the FPSMR.
6. Write the FDSR.

7. Initialize the required values for this FCC in its parameter RAM.
8. Clear out any current events in FCCE, as needed.
9. Write the FCCM register to enable the interrupts in the FCCE register.
10. Write the SCPRR_H to configure the FCC interrupt priority.
11. Clear out any current interrupts in the SIPNR_L, if preferred.
12. Write the SIMR_L to enable interrupts to the CP interrupt controller.
13. Issue an INIT TX AND RX PARAMETERS command (with the correct protocol number).
14. Set GFMR[ENT] and GFMR[ENR].

The first RxBD's empty bit must be set before the INIT RX COMMAND. However TxBDs can have their ready bits set at any time. Notice that the CPCR does not need to be accessed after a power-on reset until an FCC is to be used. An FCC should be disabled and reenabled after any dynamic change in its parallel I/O ports or serial channel physical interface configuration. A full reset using CPCR[RST] is a comprehensive reset that also can be used.

29.10 FCC Interrupt Handling

The following describes what usually occurs within an FCC interrupt handler:

1. When an interrupt occurs, read FCCE to determine interrupt sources. FCCE bits to be handled in this interrupt handler are normally cleared at this time.
2. Process the TxBDs to reuse them if the FCCE[TX,TXE] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the FCC. Thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set.
3. Extract data from the RxBD if FCCE[RX, RXB, or RXF] is set. If the receive speed is fast or the interrupt delay is long, the FCC may have received more than one receive buffer. Thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the FCC prefetches BDs, the BD table must be big enough such that always there will be another empty BD to prefetch.
4. Clear FCCE.
5. Continue normal execution.

29.10.1 FCC Transmit Errors

There are four errors in the FCC transmitter that make it necessary for software to act on the transmitter before correct operation can continue. The errors are as follows:

- CTS-lost indication in HDLC transmitter (use re-initialization procedure)
- Underrun in any of the serial FCC transmitter protocols (use re-initialization procedure)
- Late collision in fast Ethernet transmitter (use recovery or re-initialization procedure)
- Expiration of retry limit in fast Ethernet (use recovery or re-initialization procedure)

In addition to status bits in the current TxBD, these errors are reported through the TXE event bit in the FCCE as a convenience for the user to implement error handling. TXE is a safe indication that a recovery or re-initialization procedure must be started.

29.10.1.1 Re-Initialization Procedure

The following steps are required for re-initialization:

1. Disable the FCC transmission by clearing GFMR[ENT].
2. Remember the TBPTR value taken from the FCC parameter RAM.
3. Issue an “INIT TX PARAMS” command using the CPCR.
4. Restore the remembered TBPTR into the FCC parameter RAM.
5. Adjust TxBD handling as described in Section 28.10.1.3.
6. Enable FCC transmission by setting GFMR[ENT].

29.10.1.2 Recovery Sequence

The following steps are required for recovery:

1. Determine which BD is to be transmitted next and, if necessary, modify BDs.
2. Modify TBPTR field in Parameter RAM to point to next BD (if necessary).
3. Issue a “RESTART TX” command using the CPCR.

29.10.1.3 Adjusting Transmitter BD Handling

When a TXE event occurs, the TBPTR may already point beyond BDs still marked as ready due to internal pipelining. If the TBPTR is not adjusted, these BDs would be skipped while still being marked as ready. Software must determine if these BDs should be retransmitted or if they should be skipped, depending on the protocol and application needs. This requires the following steps:

1. From the current TBPTR value, search backwards over all (if any) BDs still marked as ready to find the first BD that has not been closed by the CPM. The search process should stop if the BD to be checked next is not ready or if it is the most recent BD marked as ready by the CPU transmit software. This is to avoid an endless loop in case the CPU software fills the BD ring completely.
2. A) For skipping BDs, manually close all BDs from the BD just found up to and including the BD just before TBPTR. Leave the TBPTR value untouched.

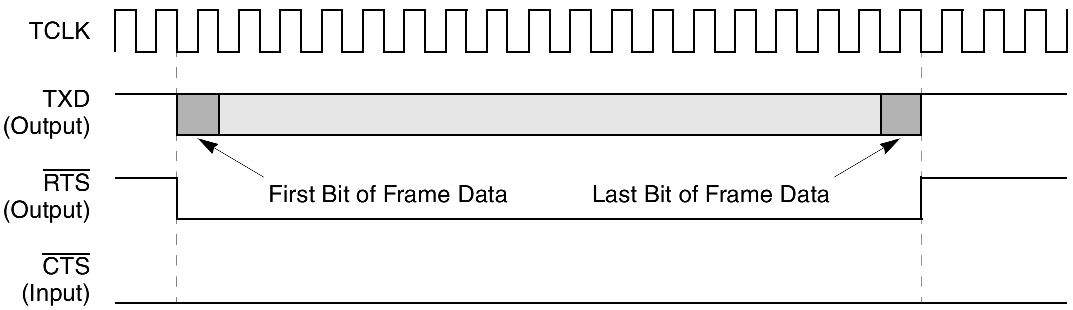
   B) For retransmitting BDs, change the TBPTR value to point to the BD just found.

29.11 FCC Timing Control

When GFMR[DIAG] is programmed to normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are automatically controlled by the FCC. GFMR[TCI] is assumed to be cleared, which implies normal transmit clock operation.

$\overline{\text{RTS}}$ is asserted when FCC has data to transmit in the transmit FIFO and a falling transmit clock occurs. At this point, the FCC begins sending the data, once the appropriate conditions occur on $\overline{\text{CTS}}$. In all cases, the first bit of data is the start of the opening flag, or sync pattern.

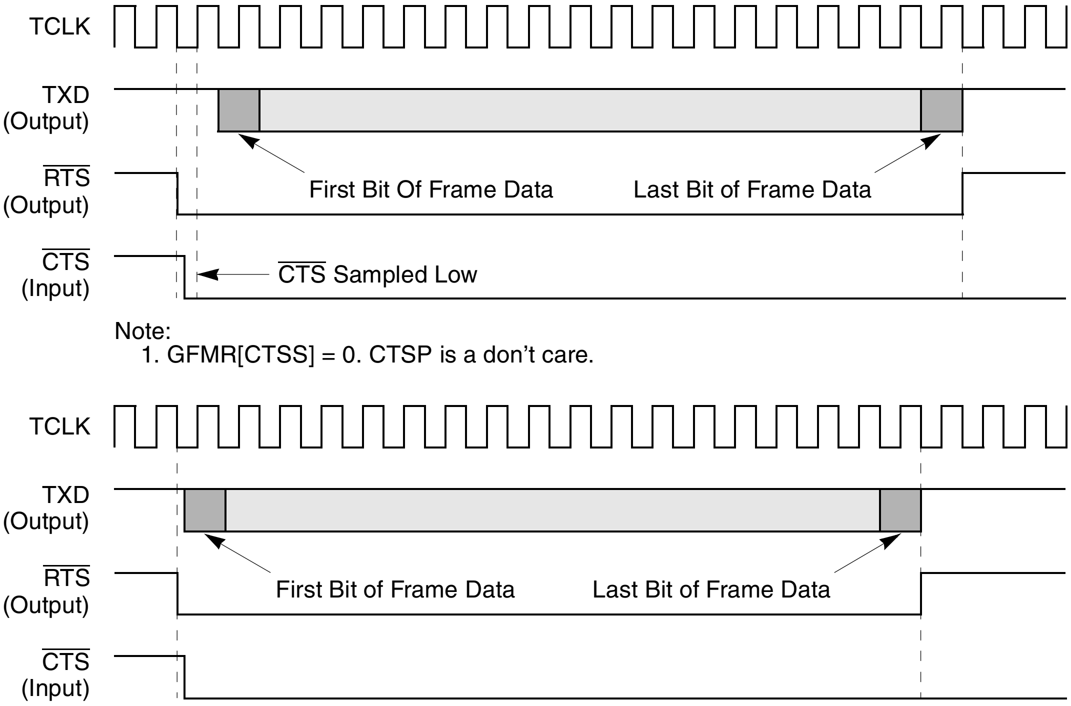
Figure 29-9 shows that the delay between RTS and data is 0 bit times, regardless of the setting of GFMR[CTSS]. This operation assumes that CTS is either already asserted to the FCC or is reprogrammed to be a parallel I/O line, in which case the CTS signal to the FCC is always asserted. RTS is negated 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 29-9. Output Delay from RTS Asserted

If CTS is not already asserted when RTS is asserted, the delays to the first bit of data depend on when CTS is asserted. Figure 29-10 shows that the delay between CTS and the data can be approximately 0.5 to 1 bit times or no delay, depending on GFMR[CTSS].



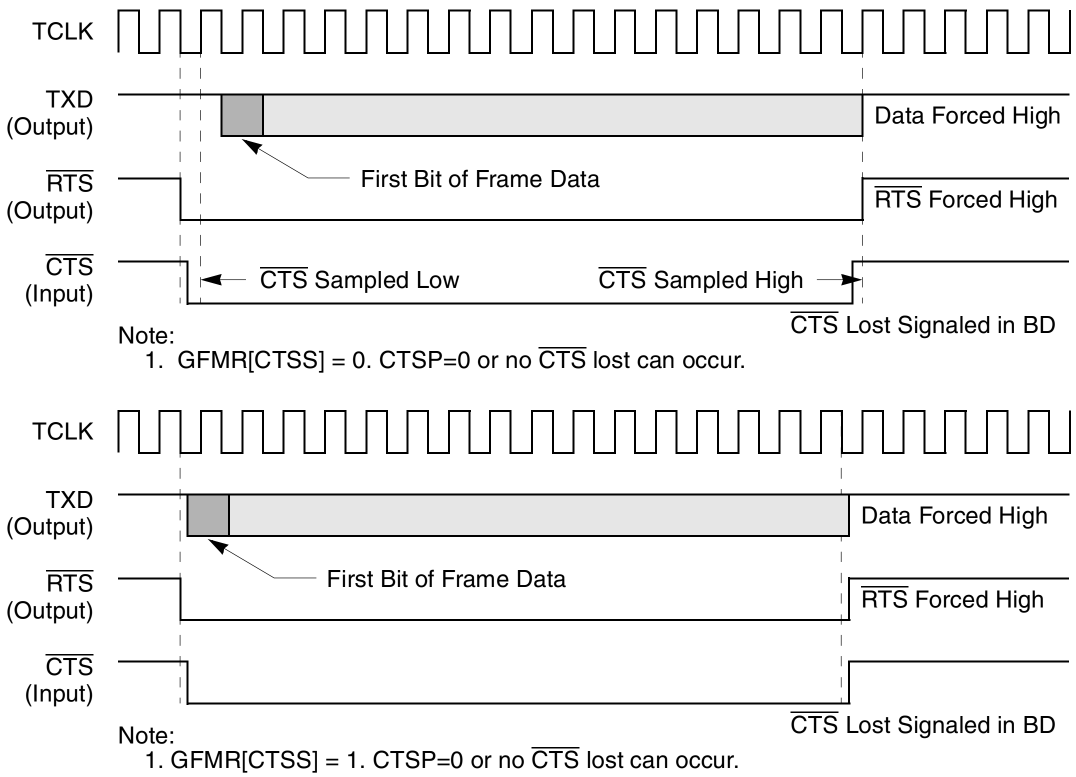
Note:
1. GFMR[CTSS] = 0. CTSP is a don't care.

Note:
1. GFMR[CTSS] = 1. CTSP is a don't care.

Figure 29-10. Output Delay from CTS Asserted

If it is programmed to envelope the data, CTS must remain asserted during frame transmission or a CTS lost error occurs. The negation of CTS forces RTS high and the transmit data to the idle state. If

GFMR[CTSS] = 0, the FCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized. Otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 29-11.



Note:
1. GFMR[CTSS] = 0. CTSP=0 or no $\overline{\text{CTS}}$ lost can occur.

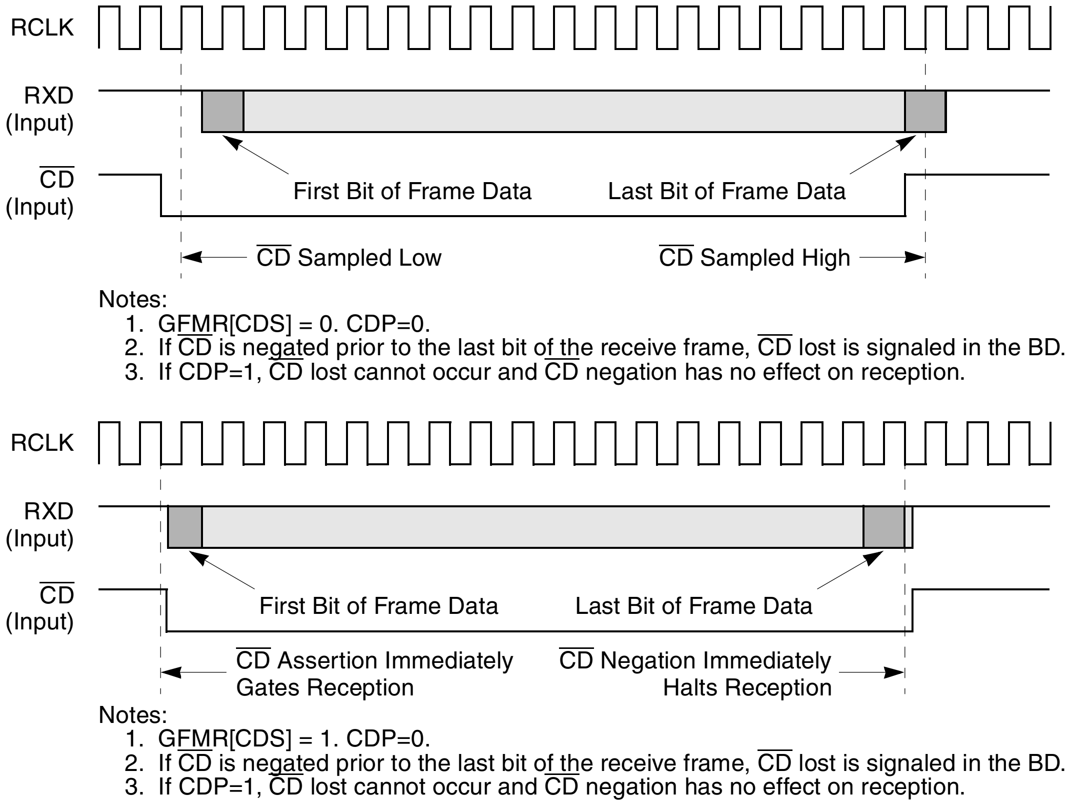
Note:
1. GFMR[CTSS] = 1. CTSP=0 or no $\overline{\text{CTS}}$ lost can occur.

Figure 29-11. $\overline{\text{CTS}}$ Lost

NOTE

If GFMR[CTSS] = 1, all $\overline{\text{CTS}}$ transitions must occur while the transmit clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as Figure 29-12 shows. If GFMR[CDS] = 0, $\overline{\text{CD}}$ is sampled on the rising receive clock edge before data is received. If GFMR[CDS] = 1, $\overline{\text{CD}}$ transitions immediately cause data to be gated into the receiver.


RCLK
RXD
(Input)
$\overline{\text{CD}}$
(Input)
First Bit of Frame Data
Last Bit of Frame Data
$\overline{\text{CD}}$ Sampled Low
$\overline{\text{CD}}$ Sampled High
RCLK
RXD
(Input)
$\overline{\text{CD}}$
(Input)
First Bit of Frame Data
Last Bit of Frame Data
$\overline{\text{CD}}$ Assertion Immediately Gates Reception
$\overline{\text{CD}}$ Negation Immediately Halts Reception

Notes:
1. GFMR[CDS] = 0. CDP=0.
2. If $\overline{\text{CD}}$ is negated prior to the last bit of the receive frame, $\overline{\text{CD}}$ lost is signaled in the BD.
3. If CDP=1, $\overline{\text{CD}}$ lost cannot occur and $\overline{\text{CD}}$ negation has no effect on reception.

Notes:
1. GFMR[CDS] = 1. CDP=0.
2. If $\overline{\text{CD}}$ is negated prior to the last bit of the receive frame, $\overline{\text{CD}}$ lost is signaled in the BD.
3. If CDP=1, $\overline{\text{CD}}$ lost cannot occur and $\overline{\text{CD}}$ negation has no effect on reception.

Figure 29-12. Using $\overline{\text{CD}}$ to Control Reception

If it is programmed to envelope data, $\overline{\text{CD}}$ must remain asserted during frame transmission or a $\overline{\text{CD}}$ lost error occurs. The negation of $\overline{\text{CD}}$ terminates reception. If [CDS] = 0, $\overline{\text{CD}}$ must be sampled by the FCC before a $\overline{\text{CD}}$ lost is recognized. Otherwise, the negation of $\overline{\text{CD}}$ immediately causes the $\overline{\text{CD}}$ lost condition.

NOTE

If GFMR[CDS] = 1, all $\overline{\text{CD}}$ transitions must occur while the receive clock is low.

29.12 Disabling the FCCs On-the-Fly

Unused FCCs can be temporarily disabled. In this case, a operation sequence is followed that ensures that any buffers in use are closed properly and that new data is transferred to or from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description states that dynamic changes are allowed, the following sequences are not required and the register or bit may be changed immediately. In all other cases, the sequence should be used.

Modifying parameter RAM does not require the FCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

29.12.1 FCC Transmitter Full Sequence

For the FCC transmitter, the full disable and enable sequence is as follows:

1. Issue the STOP TRANSMIT command. This is recommended if the FCC is currently transmitting data because it stops transmission in an orderly way. If the FCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR is overwritten by the user or the INIT TX PARAMETERS command is executed, this command is not required.
2. Clear GFMR[ENT]. This disables the FCC transmitter and puts it in a reset state.
3. Make changes. The user can modify FCC transmit parameters, including the parameter RAM. To switch protocols or restore the FCC transmit parameters to their initial state, the INIT TX PARAMETERS command must be issued.
4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.
5. Set GFMR[ENT]. Transmission begins using the TxBD that the TBPTR points to as soon as TxBD[R] = 1.

29.12.2 FCC Transmitter Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENT].
2. Issue the INIT TX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENT].

29.12.3 FCC Receiver Full Sequence

The full disable and enable sequence for the receiver is as follows:

1. Clear GFMR[ENR]. Reception is aborted immediately, which disables the receiver of the FCC and puts it in a reset state.
2. Make changes. The user can modify the FCC receive parameters, including the parameter RAM. If the user prefers to switch protocols or restore the FCC receive parameters to their initial state, the INIT RX PARAMETERS command must be issued.
3. Issue the ENTER HUNT MODE command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set GFMR[ENR]. Reception begins immediately using the RxBD that the RBPTR points to if RxBD[E] = 1.

29.12.4 FCC Receiver Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the receive parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENR].

2. Issue the INIT RX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENR].

29.12.5 Switching Protocols

A user can switch the protocol that the FCC is executing (HDLC) without resetting the board or affecting any other FCC by taking the following steps:

1. Clear GFMR[ENT] and GFMR[ENR].
2. Issue the INIT TX AND RX PARAMETERS command. This command initializes both transmit and receive parameters. Additional changes can be made in the GFMR to change the protocol.
3. Set GFMR[ENT] and GFMR[ENR]. The FCC is enabled with the new protocol.

29.13 Saving Power

Clearing an FCC’s ENT and ENR bits minimizes its power consumption.

Chapter 30
ATM Controller and
AAL0, AAL1, and AAL5 Protocols

NOTE

> The functionality described in this chapter is not available on the MPC8248 and the MPC8247. On the MPC8272, it is supported only on FCC1.

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 2) for both master and slave modes. It performs segmentation and reassembly (SAR) functions of AAL5, AAL2, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.

For each virtual channel (VC), the controller's ATM pace control (APC) unit generates a cell transmission rate to implement constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR), unspecified bit rate (UBR) or UBR+ traffic. To regulate VBR traffic, the APC unit performs a continuous-state leaky bucket algorithm. The APC unit also uses up to eight priority levels to prioritize real-time ATM channels, such as CBR and real-time VBR, over non-real-time ATM channels such as VBR, ABR and UBR.

The ATM controller performs the ATM forum (UNI-4.0) ABR flow control. To perform feedback rate adaptation, it supports forward and backward resource management (RM) cell generation and ATM forum floating-point calculation. ABR flow control is implemented in hardware and firmware (without software intervention) to prevent potential delays during backward RM cell processing and feedback rate adaptation.

The MPC8272 supports a special mode for ATM/TDM interworking. The CPM performs automatic data forwarding without core intervention.

The MPC8272 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

30.1 Features

The ATM controller has the following features:

- Full duplex segmentation and reassembly at 155 Mbps
- UTOPIA level 2 master and slave modes (8 bit)
- AAL5, AAL1, AAL2, AAL0 protocols

- Up to 255 active VCs internally, and up to 64K VCs using external memory
- TM 4.0 CBR, VBR, UBR, UBR+ traffic types
- VBR type 1 and 2 traffic using leaky buckets (GCRA)
- TM 4.0 ABR flow control (EFCI and ER)
- Idle/unassign cells screening/transmission option
- External and internal rate transmit modes
- Special mode for ATM-to-port 2 or ATM-to-ATM data forwarding
- CLP and congestion indication marking
- User-defined cells up to 65 bytes
- Separate TxBD and RxBD tables for each virtual channel (VC)
- Special mode of global free buffer pools for dynamic and efficient memory allocation with early packet discard (EPD) support
- Interrupt report per channel using four priority interrupt queues
- Compliant with ATMF UNI 4.0 and ITU specification
- AAL5 cell format
  - Reassembly
    - Reassemble PDU directly to external memory
    - CRC32 check
    - CLP and congestion report
    - CPCS_UU, CPI, and length check
    - Abort message report
  - Segmentation
    - Segment PDU directly from external memory
    - Performs PDU padding
    - CRC32 generation
    - Automatic last cell marking
    - Automatic CPCS_UU, CPI, and length insertion
    - Abort message option
- AAL1 cell format
  - Reassembly
    - Reassemble PDU directly to external memory
    - Support for partially filled cells (configurable on a per-VC basis)
    - Sequence number check
    - Sequence number protection (CRC-3 and parity) check
  - Segmentation
    - Segment PDU directly from external memory
    - Partially filled cells support (configurable on a per-VC basis)

- Sequence number generation
- Sequence number protection (CRC-3 and even parity) generation

  - Structured AAL1 cell format
    - Automatic synchronization using the structured pointer during reassembly
    - Structured pointer generation during segmentation
  - Unstructured AAL1 cell format
    - Clock recovery using external SRTS (synchronous residual time stamp) logic during reassembly
    - SRTS generation using external logic during segmentation
- AAL0 format
  - Receive
    - Whole cell is put in memory
    - CRC10 pass/fail indication
  - Transmit
    - Reads a whole cell from memory
    - CRC10 insertion option
- AAL2 format
  - Refer to Chapter 31, “AAL2 Protocol.”
- 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 2 supports 8 bits 25/50 MHz
    - Supports UTOPIA master and slave modes
    - Supports cell-level handshake
    - Supports multi-PHY polling mode
- ATM pace control (APC) unit
  - Peak cell rate pacing on a per-VC basis
  - Peak-and-sustain cell rate pacing using GCRA on a per-VC basis
  - Peak-and-minimum cell rate pacing on a per-VC basis
  - Up to eight priority levels
  - Fully managed by CP with no host intervention
- Available bit rate (ABR)
  - Performs ATMF UNI 4.0 ABR flow control on a per-VC basis

  - Automatic forward-RM, backward-RM cells generation
  - Automatic feedback rate adaptation
  - Support for EFCI (explicit forward congestion indication) and ER (explicit rate)
  - RM cell floating-point calculations
  - Fully managed by CP with no host intervention
- Receive address look-up mechanism
  - Two modes of address look-up are supported.
    - External CAM
    - Address compression
- OAM (operations and maintenance) cells
  - OAM filtering according to PTI field and reserved VCI field
  - Raw cell queues for transmission and reception
  - CRC-10 generation/check
  - Performance monitoring support
    - Support up to 64 bidirectional block tests simultaneously
    - Automatic FMC and BRC cell generation and termination
    - User transmit cell_{0+1} count
    - User transmit cell_0 count
    - PM cells time stamp insertion
    - Block error detection code (BEDC_{0+1}) generation/check
    - Total receive cell_{0+1} count
    - Total receive cell_0 count
  - Specifying channel code for F5 OAM cells
- ATM layer statistic gathering on a per PHY basis
  - UTOPIA receiver error cells count (Rx parity error or short/long cells error)
  - Misinserted cell count
  - CRC-10 error cells count (ABR flow only)
- Memory management
  - RxBD table per VC with option of global free buffer pool for AAL5
  - TxBD table per VC

30.2 ATM Controller Overview

The following sections provide an overview of the transmitter and receiver portions of the ATM controller.

30.2.1 Transmitter Overview

Before the transmitter is enabled, the host must initialize the MPC8272 and create the transmit data structure, described in Section 30.10, “ATM Memory Structure.” When data is ready for transmission, the host arranges the BD table and writes the pointer of the first BD in the transmit connection table (TCT).

The host issues an ATM TRANSMIT command, which inserts the current channel to the ATM pace control (APC) unit. The APC unit controls the ATM traffic of the transmitter. It reads the traffic parameters of each channel and divides the total bandwidth among them. The APC unit can pace the peak cell rate, peak-and-sustain cell rate (GCRA traffic) or peak-and-minimum cell rate traffic. The APC implements up to eight priority levels for servicing real-time channels before non-real-time channels.

The transmitter ATM cell is 53–65 bytes and includes 4 bytes of ATM cell header, a 1-byte HEC, and 48 bytes of payload. The HEC is a constant taken from FDSR*x*[0–15] when using UTOPIA 16 and from FDSR*x*[8–15] when using UTOPIA 8; see Section 29.4, "FCC Data Synchronization Registers (FDSRx)." User-defined cells (UDC mode) include an extra header of 1–12 bytes with an optional HEC octet. Cell transfers use the UTOPIA level 2, cell-level handshake.

Transmission starts when the APC schedules a channel. According to the channel code, the ATM controller reads the channel's entry in the TCT and opens the first BD for transmission. In auto-VC-off mode, the APC automatically deactivates the current channel when no buffer is ready to transmit. In this case, a new ATM TRANSMIT command is needed for transmission of the VC to resume.

30.2.1.1 AAL5 Transmitter Overview

The transmitter reads 48 bytes from the external buffer, adds the cell header, and sends the cell through the UTOPIA interface. The transmitter adds any padding needed and appends the AAL5 trailer in the last cell of the AAL5 frame. The trailer consists of CPCS-UU+CPI, data length, and CRC-32 as defined in ITU I.363. The CPCS-UU+CPI (2-byte entry) can be specified by the user or optionally cleared by the transmitter; see Section 30.10.2.3, "Transmit Connection Table (TCT)." The transmitter identifies the last cell of the AAL5 message by setting the last (L) indication bit in the PTI field of the cell header. An interrupt may be generated to indicate the end of the frame.

When the transmission of the current frame ends and no additional valid buffers are in the BD table, the transmit process ends. The transmitter keeps polling the BD table every time this channel is scheduled to transmit. Note that a buffer-not-ready indication during frame transmission aborts the frame transfer.

30.2.1.2 AAL1 Transmitter Overview

The MPC8272 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 MPC8272 supports synchronous residual time stamp (SRTS) generation using external PLL. If this mode is enabled, the MPC8272 reads the SRTS code from the external logic and inserts it into four outgoing cells.

For the structured format, the transmitter reads 47 or 46 bytes from the external buffer and inserts them into the AAL1 user data field. The CP generates the AAL1 PDU header and inserts it into the cell. The header consists of the SN, SNP, and the structured pointer.

The MPC8272 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.

30.2.1.3 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.

30.2.1.4 AAL2 Transmitter Overview

Refer to Section 31.3.1, "Transmitter Overview."

30.2.1.5 Transmit External Rate and Internal Rate Modes

The ATM controller supports the following two rate modes:

- External rate mode—The total transmission rate is determined by the PHY transmission rate. The FCC sends cells to keep the PHY FIFOs full; the FCC inserts idle/unassign cells to maintain the transmission rate.
- Internal rate mode—The total transmission rate is determined by the FCC internal rate timers. In this mode, the FCC does not insert idle/unassign cells. The internal rate mechanism supports up to 4 different rates. Each PHY has its own FTIRR, described in Section 30.13.5, "FCC Transmit Internal Rate Registers (FTIRR1)." The FTIRR includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer expires. When using internal rate mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers.

30.2.2 Receiver Overview

Before the receiver is enabled, the host must initialize the MPC8272 and create the receive data structure described in Section 30.10, "ATM Memory Structure." The host arranges a BD table for each ATM channel. Buffers for each connection can be statically allocated (that is, each BD in the BD table is associated with a fixed buffer location) or in the case of AAL5, can be fetched by the CP from a global free buffer pool. See Section 30.10.5, "ATM Controller Buffer Descriptors (BDs)."

The receiver ATM cell size is 53–65 bytes. The cell includes: 4 bytes ATM cell header, 1 byte HEC, which can be checked by setting FPSMR[HECC], 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 2, cell-level handshake.

Reception starts when the PHY asserts the receive cell available signal (RxCLAV) to indicate that the PHY has a complete cell in its receive FIFO. The receiver reads a complete cell from the UTOPIA interface and translates the header address (VP/VC) to a channel code by performing an address look-up. If no matches are found, the cell is discarded and the user-network interface (UNI) statistics tables are updated. The receiver uses the channel code to read the channel parameters from the receive connection table (RCT).

30.2.2.1 AAL5 Receiver Overview

The receiver copies the 48-byte cell payload to the external buffer and calculates CRC-32 on the entire CPCS-PDU. When the last AAL5 cell arrives, the receiver checks the length, CRC-32, and

CPCS-UU+CPI fields and sets the corresponding RxBD status bits. An interrupt may be generated to one of the four interrupt queues. The receiver copies the last cell to memory including the padding and the AAL5 trailer. The CPCS-UU+CPI (16-bit entry) may be read directly from the AAL5 trailer.

The ATM controller monitors the CLP and CNG state of the incoming cells. When the message is closed, these events set RxBD[CLP] and RxBD[CNG].

When no buffer is ready to receive cells (busy state), the receiver switches to hunt state and drops all cells associated with the current frame (partial packet discard). The receiver tries to open new buffers for cell reception only after the last cell of the discarded AAL5 frame arrives.

30.2.2.2 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 MPC8272 supports SRTS clock recovery using an external PLL. In this mode, the MPC8272 tracks the SRTS from the four incoming cells and writes the SRTS code to external logic.

In the unstructured format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and starts receiving. If an SN mismatch is detected, the receiver closes the RxBD, sets the sequence number error flag (RxBD[SNE]), and switches to hunt state, where it stays until a cell with a valid SN field is received.

For the structured format, 47 or 46 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of SN and SNP, is checked and the PDU status is written to the BD.

In the structured format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer to gain synchronization. When one arrives, the receiver leaves the hunt state and starts receiving. Then the receiver opens a new buffer. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer. If an SN mismatch is detected, the ATM receiver closes the current RxBD, sets RxBD[SNE], and returns to the hunt state. The receiver then waits for a cell with a valid structured pointer to regain synchronization.

The MPC8272 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.

30.2.2.3 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.

30.2.2.4 AAL2 Receiver Overview

Refer to Section 31.4.1, “Receiver Overview.”

30.2.3 Performance Monitoring

The ATM controller supports performance monitoring testing according to ITU I.610. When performance monitoring is enabled, the ATM controller automatically generates and terminates FMCs (forward monitoring cells) and BRCs (backward reporting cells). See Section 30.6.6, “Performance Monitoring.”

30.2.4 ABR Flow Control

When AAL5-ABR is enabled, the ATM controller implements the ATM Forum TM 4.0 available-bit-rate flow. It automatically inserts forward- and backward-RM cells into the user cell stream and adjusts the transmission rate according to the backwards RM cell feedback; see Section 30.10.2.2.2, “AAL5-ABR Protocol-Specific RCT.” The ABR flow is controlled on a per-VC basis.

30.3 ATM Pace Control (APC) Unit

The ATM pace control (APC) unit schedules the ATM channels for transmitting. While performing this task, the APC unit uses the following parameters:

- Frequency (bandwidth) of each ATM channel
- ATM traffic pacing—Peak cell rate (PCR), sustain cell rate (SCR), and minimum rate (MCR)
- Priority level—Real-time channels (CBR or VBR-RT) are scheduled at high-priority levels; non-real-time channels (VBR-NRT, ABR, UBR) are scheduled at low-priority levels. Up to eight priority levels are available.

30.3.1 APC Modes and ATM Service Types

The ATM Forum (http://www.atmforum.com) defines the service types described in Table 30-1.

Table 30-1. ATM Service Types

Service Type	Cell Rate Pacing	Real-Time/ Non-Real-Time	Relative Priority
CBR	PCR	RT	1 (highest)
VBR-RT	PCR, SCR (peak-and-sustain)	RT	2
VBR-NRT	PCR, SCR (peak-and-sustain)	NRT	3
ABR[1]	PCR	NRT	4
UBR+	PCR, MCR (peak-and-minimum)	NRT	5
UBR	PCR	NRT	6 (lowest)

[1] When ABR flow control is active, the CP automatically adapts the APC parameters PCR, PCR_FRACTION. These parameters function as the channel’s allowed cell rate (ACR).

For information about cell rate pacing, see Section 30.3.5, “ATM Traffic Type.” For information about prioritization, see Section 30.3.6, “Determining the Priority of an ATM Channel.”

30.3.2 APC Unit Scheduling Mechanism

The APC unit consists of an APC data structure in the dual-port RAM for each PHY and a special scheduling algorithm performed by the CP. Each PHY's APC data structure includes three elements: an APC parameter table, an APC priority table, and cell transmission scheduling tables for each priority level. (See Section 30.10.4, "APC Data Structure.")

Each PHY's APC parameter table holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The priority table holds pointers that define the location and size of each priority level's scheduling table.

Each scheduling table is divided into time slots, as shown in Figure 30-1. The user determines the number of ATM cells to be sent each time slot (cells per slot). After a channel is sent, it is removed from the current time slot and advanced to a future time slot according to the channel's assigned traffic rate (specified in time slots). The PCR parameter in the TCT, or the SCR or MCR parameters in the TCT extension (TCTE) determine the channel's actual rate.

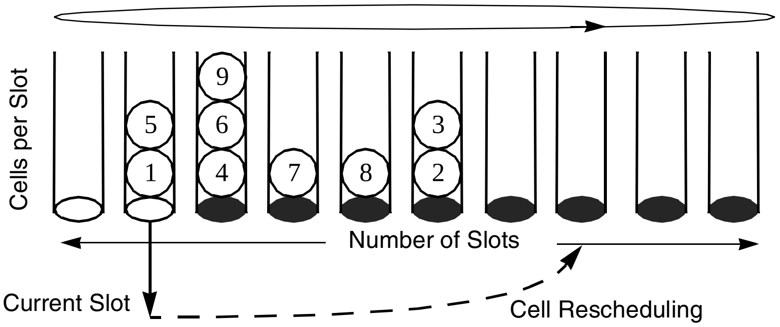


Figure 30-1. APC Scheduling Table Mechanism

Each 2-byte time-slot entry points to one ATM channel. Additional channels scheduled to transmit in the same slot are linked to each other using the APC linked-channel field in the TCT. The linked list is not limited; however, if the number of channels for the current slot exceeds the cells per slot parameter (CPS), the extra channels are sent in subsequent time slots. (The rescheduling of extra channels is based on the original slot to maintain each channel's pace.)

Note that a channel can appear only once in the scheduling table at a given time, because each channel has only one APC linked-channel field.

30.3.3 Determining the Scheduling Table Size

The following sections describe how to determine the number of cells sent per time slot and the total number of slots needed in a scheduling table.

30.3.3.1 Determining the Cells Per Slot (CPS) in a Scheduling Table

The number of cells sent per time slot is determined by the channel with the maximum bit rate; see equation A. The maximum bit rate is achieved when a channel is rescheduled to the next slot. For example, if the line rate is 155.52 Mbps and there are eight cells per slot, equation A yields a maximum VC rate of 19.44 Mbps.

$$\text{(A)} \qquad \text{Max Bit Rate} = \frac{\text{Line Rate}}{\text{Cells per Slot}}$$

Note that a channel can appear only once per time slot; thus, 19.44 Mbps = 155.52Mbps/8.

The cells per slot parameter (CPS) affects the cell delay variation (CDV). Because the APC unit does not put cells in a definite order within each time slot (LIFO—last-in/first-out implementation), as CPS increases, the CDV increases. However as CPS decreases, the size of the scheduling table in the dual-port RAM increases and more CPM bandwidth is required.

30.3.3.2 Determining the Number of Slots in a Scheduling Table

The number of time slots in a scheduling table is determined by the channel with the minimum bit rate; see equation B. The minimum bit rate is achieved when the channel reschedules only once in a whole table scan. (The maximum schedule advance allowed is equal to number_of_slots-1.) For example, if the line rate is 155.52 Mbps, the minimum bit rate is 32 Kbps and the CPS is 4, then, according to equation B, the number of slots should be 1,216.

NOTE

> The following APC configuration is not recommended for values outside the following ranges (PCR = peak cell rate, PCRF = peak cell rate fraction, NOS = number of slots):
>
> PCR = 1 and PCRF = 0
>
> PCR = NOS - 1 and PCRF = 0

$$\text{(B)} \qquad \text{Min Bit Rate} = \frac{\text{Line Rate}}{(\text{number\_of\_slots} - 1) \times \text{Cells per Slot}}$$

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:

$$\text{Max Bit Rate} = (155.52\ \text{Mbps})/8 = 19.44\ \text{Mbps}$$

$$\text{Min Bit Rate} = (155.52\ \text{Mbps})/(1024 \times 8) = 18.98\ \text{Kbps}$$

30.3.4 Determining the Time-Slot Scheduling Rate of a Channel

The time-slot scheduling rate of each ATM channel is defined by equation C. The resulting number of APC slots is written in either TCT[PCR], TCTE[SCR] or TCTE[MCR], depending on the traffic type.

$$\text{(C)} \qquad \text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}}$$

30.3.5 ATM Traffic Type

The APC uses the cell rate pacing parameters (PCR, SCR, and MCR) to generate CBR, VBR, ABR, UBR+, and UBR traffic. The user determines the kind of traffic that is generated per VC by writing to TCT[ATT] (ATM traffic type); see Section 30.10.2.3, “Transmit Connection Table (TCT).”

30.3.5.1 Peak Cell Rate Traffic Type

When the peak cell rate traffic type is selected, the APC schedules channels to transmit according to the PCR and PCR_FRACTION traffic parameters. Other traffic parameters do not apply to this traffic type.

30.3.5.2 Determining the PCR Traffic Type Parameters

Suppose a VC uses 15.66 Mbps of the total 155.52 Mbps and CPS = 8. Equation C yields:

$$\text{PCR [slots]} = (155.52\text{ Mbps})/(15.66\text{ Mbps} \times 8) = 1.241$$

The resulting number of slots is written into TCT[PCR] and TCT[PCR_FRACTION]. Because PCR_FRACTION is in units of 1/256 slots, the fraction must be converted as follows:

$$1.241 = 1 + 0.241 \times 256/256 = 1 + 61.79/256 \sim 1 + 62/256$$

$$\text{PCR} = 1 \qquad \text{PCR\_FRACTION} = 62$$

30.3.5.3 Peak and Sustain Traffic Type (VBR)

Variable bit rate (VBR) traffic can burst at the peak cell rate as long as the long-term average rate does not exceed the sustainable cell rate. To support VBR channels, the APC implements the GCRA (generic cell rate algorithm) using three parameters—the peak cell rate (PCR), the sustained cell rate (SCR), and burst tolerance (BT), as shown in Figure 30-2. (The GCRA is also known as the leaky bucket algorithm.)

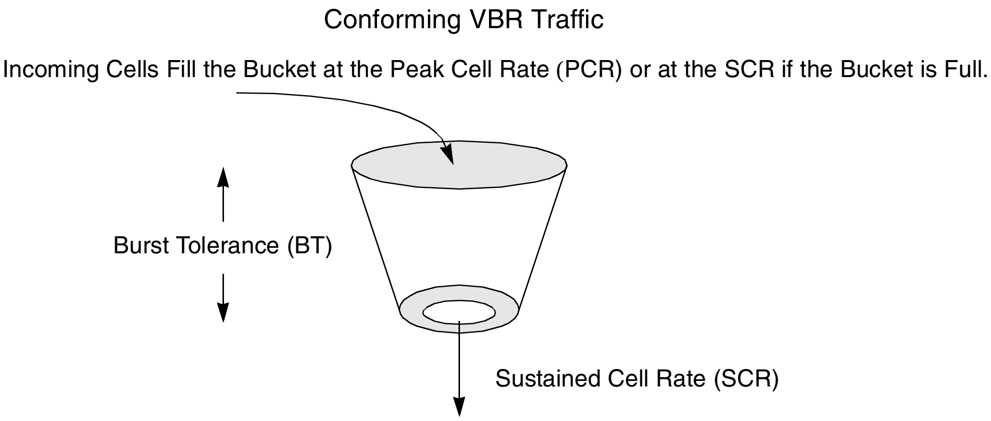

Conforming VBR Traffic
Incoming Cells Fill the Bucket at the Peak Cell Rate (PCR) or at the SCR if the Bucket is Full.
Burst Tolerance (BT)
Sustained Cell Rate (SCR)

Figure 30-2. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

When a VBR channel is activated, it bursts at the peak cell rate (PCR) until reaching its initial burst tolerance (BT), which is the buffer length the network allocated for this VC. When the burst limit is reached, the APC reduces the VC’s scheduling rate to the sustained cell rate (SCR). The VC continues sending at SCR as long as TxBDs are ready. However, as each SCR time allotment elapses with no TxBD ready to send, the APC grants the VC a credit for bursting at the peak cell rate (PCR). (Gaining credit implies that the buffer at the switch is not full and can tolerate a burst transmission.) If a TxBD becomes ready, the APC schedules the VC to burst at the PCR as long as credit remains. When the burst credit ends (the network's UPC leaky bucket reaches its limit), the APC schedules the VC according to SCR.

30.3.5.3.1 Example for Using VBR Traffic Parameters

Suppose the traffic parameters of a VBR channel are PCR = 6 Mbps, SCR = 2 Mbps, MBS (maximum burst size) = 1000 cells, and CPS = 8.

Equation C (see Section 30.3.4, "Determining the Time-Slot Scheduling Rate of a Channel") yields the APC parameters, PCR, PCR_FRACTION, SCR, and SCR_FRACTION, which the user writes to the channel's TCT.

$$\text{PCR [slots]} = (155.52\ \text{Mbps})/(6\ \text{Mbps} \times 8) = 3.24$$

$$3.24 = 3 + 0.24 \times 256/256 = 3 + 61.44/256 \sim 3 + 62/256$$

$$\text{PCR} = 3 \qquad \text{PCR\_FRACTION} = 62$$

$$\text{SCR [slots]} = (155.52\ \text{Mbps})/(2\ \text{Mbps} \times 8) = 9.72$$

$$9.72 = 9 + (0.72 \times 256/256) = 9 + 184.32/256 \sim 9 + 185/256$$

$$\text{SCR} = 9 \qquad \text{SCR\_FRACTION} = 185$$

Equation D yields the number of slots the user writes to the channel's TCT[BT].

$$\text{(D)} \qquad \begin{aligned} \text{BT [slots]} &= (\text{MBS[cells]} - 2) \times (\text{SCR[slots]} - \text{PCR[slots]}) + \text{SCR[slots]} \\ &= (1000 - 2) \times ((9+185/256) - (3+62/256)) + (9 + 185/256) \\ &= 6477 \end{aligned}$$

30.3.5.3.2 Handling the Cell Loss Priority (CLP)—VBR Type 1 and 2

The MPC8272 supports two ways to schedule VBR traffic based on the cell loss priority (CLP). When TCTE[VBR2] is cleared, CLP_{0+1} cells are scheduled by PCR or SCR according to the GCRA state. When TCTE[VBR2] is set, CLP_0 cells are still scheduled by PCR or SCR according to the GCRA state, but CLP_1 cells are always scheduled by PCR. See Section 30.10.2.3.5, "VBR Protocol-Specific TCTE."

30.3.5.4 Peak and Minimum Cell Rate Traffic Type (UBR+)

To support UBR+ channels, the APC schedules transmission according to PCR and MCR. For each priority level, the APC maintains a parameter that monitors the traffic load measured as the time-slot delay between the service pointer (pointing to the current time slot waiting transmission) and a real-time slot pointer. If the transmission delay is greater than MDA (maximum delay allowed), the APC begins scheduling channels according to the MCR parameter. If the delay, however, drops below MDA, the APC again schedules channels according to the PCR. Note that in order to guarantee a minimum cell rate for

UBR+ channels, there must be enough bandwidth to simultaneously send all possible channels at the MCR. See Section 30.10.2.3.6, "UBR+ Protocol-Specific TCTE."

30.3.6 Determining the Priority of an ATM Channel

The priority mechanism is implemented by adding priority table levels, which point to separate scheduling tables; see Section 30.10.4, "APC Data Structure." The APC flow control services the APC_LEVEL1 slots first. If there are no cells to send, the APC goes to the next priority level. The APC has up to eight priority levels with APC_LEVEL8 being the lowest. The user specifies the priority of an ATM channel when issuing the ATM TRANSMIT command; see Section 30.15, "ATM Transmit Command."

The real-time channels, CBR and VBR-RT, should be inserted in APC_LEVEL1; non-real-time channels, VBR-NRT, ABR, and UBR should be inserted in lower priority levels.

30.4 VCI/VPI Address Lookup Mechanism

The MPC8272 supports two ways to look up addresses for incoming cells:

- External CAM lookup
- Address compression

Writing to GMODE[ALM] (address-lookup-mechanism bit) in the parameter RAM selects the mechanism. Both mechanisms are described in the following sections.

30.4.1 External CAM Lookup

An external CAM is usually used when the range of VCI/VPI values varies widely or is unknown. Clearing GMODE[ALM] selects the external CAM address lookup mechanism. If there is no match in the external CAM, the cell is considered a misinserted cell. The external CAM can point to internal or external channels (channels whose connection table resides in external memory). The CAM input, shown in Figure 30-3, is the 32-bit cell address: PHY address, GFC + VPI, and VCI.

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

0 3	4 15	16 31
PHY Addr (MPHY)	GFC + VPI	VCI

Figure 30-3. External CAM Data Input Fields

The output of the CAM, shown in Figure 30-4, is a 32-bit entry (16-bit channel code and a match-status bit).

0	1 15	16 31
MS	—	Channel Code

Figure 30-4. External CAM Output Fields

The external CAM fields are described in Table 30-2.

Table 30-2. External CAM Input and Output Field Descriptions

Field	Description
PHY Addr	In multi-PHY mode, this field contains the 4 least-significant bits of the current channel's physical address. Because this CAM comparison field is limited to 4 bits, two CAM devices are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two devices. If the msb is zero, the CP accesses the CAM whose address is written in the EXT_CAM_BASE parameter in the parameter RAM; if the msb is set, the CP uses EXT_CAM1_BASE. See Section 15.4.1, "CMX Utopia Register (CMXATMR) (MPC8272 only)." In single-PHY mode, clear this field.
GFC+VPI, VCI	The GFC, VPI, and VCI of the current channel
Ch Code	Pointer to internal or external connection table
—	Reserved, should be cleared.
MS	Match status. 0 Match was found 1 Match was not found

30.4.2 Address Compression

The address compression mechanism uses two levels of address translation to help minimize the memory space needed to cover the available address range. The first level of translation (VP-level) uses a look-up table based on the 4-bit PHY address and the 12-bit virtual path identifier; the second level (VC-level) uses the 16-bit virtual channel identifier. If there is no match during address compression, the cell is considered a misinserted cell.

During the VP-level translation, VP_MASK in the ATM parameter RAM compresses an incoming cell's PHY address and VPI to create an index into the VP-level table. The VP-level table entry consists of another mask (VC_MASK) and a pointer to one of the VC-level tables (VCOFFSET). Note that the VP table should reside in the dual-port RAM.

In the VC-level translation, the VCI is compressed with the VC_MASK to generate a pointer to the VC-level table entry containing the received cell's channel code. The VC table should reside in external memory.

Figure 30-5 shows an example of address compression.

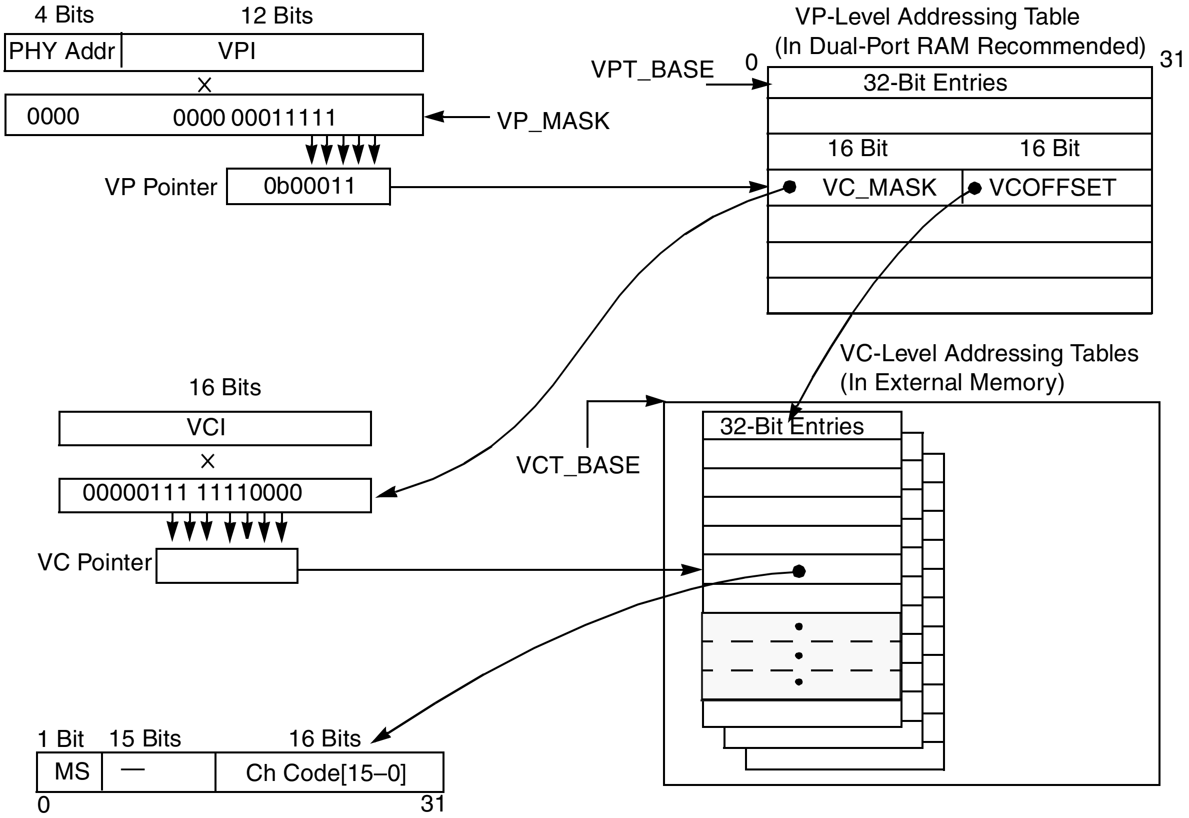


Figure 30-5. Address Compression Mechanism

Figure 30-5 shows VP_MASK selecting five VPI bits to index the VP-level table. The VP-level table entry contains the 16-bit mask (VC_MASK) and the VC-level table offset (VCOFFSET) for the next level of address mapping. The VC_MASK selects VCI bits 4–10, which are used with VCT_BASE and VCOFFSET to indicate the received cell’s channel code. Address compression field descriptions are shown in Table 30-3.

Table 30-3. Field Descriptions for Address Compression

Field	Description
PHY Addr	In multi-PHY mode, this field contains the 4 least-significant bits of the current channel’s physical address. Because this comparison field is limited to 4 bits, two sets of look-up tables are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two sets of tables. If the msb is zero, the CP accesses the tables at VPT_BASE and VCT_BASE; if the msb is set, the CP uses VPT1_BASE and VCT1_BASE. See Section 15.4.1, “CMX Utopia Register (CMXATMR) (MPC8272 only).” In single-PHY mode, clear this field.
VCI, VPI	The VCI and VPI of the current channel
Ch Code	Pointer to internal or external connection table
—	Reserved, should be cleared.
MS	Match status. 0 Match was found 1 Match was not found

30.4.2.1 VP-Level Address Compression Table (VPLT)

The size of the VP-level table depends on the number of mask bits in VP_MASK. For example, if only one PHY is available (PHY address = 0) and VPMASK = 0b11_1111_1111, VP pointer contains 10 bits and the table is 4 Kbytes. Because each VPLT entry is 4 bytes, the address of an entry is VPT_BASE+VP pointer × 4.

Each VPLT entry has two parameters:

- VC_MASK—A 16-bit VC-level mask for masking the incoming cell's VCI
- VCOFFSET—A 16-bit VC-level table offset from VC_BASE that points to the appropriate VC-level table's (VCLT) starting address. The address of the VCLT is VC_BASE + VCOFFSET × 4.

  If the VCLTs are to be placed contiguously in memory, each table's VCOFFSET depends on the size of preceding tables. Each table's size depends on the number of ones in VC_MASK. Figure 30-6 gives the general formula for determining VCOFFSET.

General Formula: VCOFFSET$_{(n+1)}$ = VCOFFSET$_{n}$ + $2^{(\text{number of ones in VC\_MASKn})}$

Figure 30-6. General VCOFFSET Formula for Contiguous VCLTs

Table 30-4 shows example VCOFFSET calculations for a VP-level table with four entries.

Table 30-4. VCOFFSET Calculation Examples for Contiguous VCLTs

VP-Level Table Entry	VC_MASK	Number of Ones in VC_MASK	VC-Level Table Size	VCOFFSET
0	0x0237	6	2^6 = 64 entries	0
1	0x0230	3	2^3 = 8 entries	64
2	0xA007	5	2^5 = 32 entries	64 + 8 = 72
3	x	x	x	72 + 32 = 104

The MPC8272 can check that all unallocated bits of the PHY + VPI are 0 by setting GMODE[CUAB] (check unallocated bits) in the parameter RAM. If they are not, the cell is considered a misinserted cell.

Table 30-5 gives an example of VP-level table entry address calculation.

Table 30-5. VP-Level Table Entry Address Calculation Example

VPT_BASE	VP-Level Table Size	VP_MASK	Phy+VPI	VP Pointer	VP Entry Address
0x0024_0000	64 entries	0x0237	0x0011	0x09	VP base = 0x240000 0x09 x 4 = 0x000024 0x240024

Figure 30-7 shows the VP pointer address compression from Table 30-5.

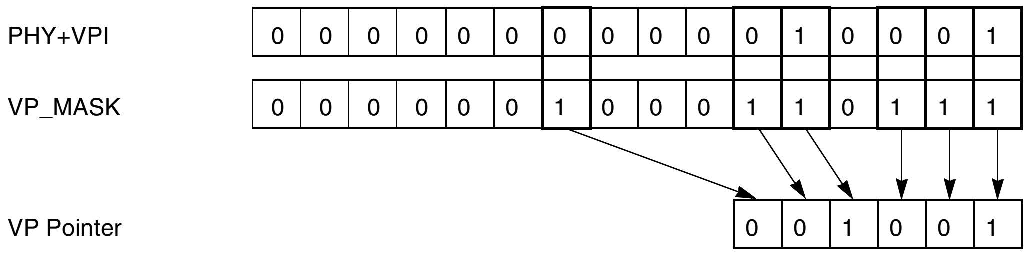


Figure 30-7. VP Pointer Address Compression

30.4.2.2 VC-Level Address Compression Tables (VCLTs)

Each VPLT entry points to a single VCLT. Like the VPLT, the size of each VCLT depends on VC_MASK. Because the VCLT contains word entries, if VC_MASK = 0b11_1111_1111, the table is 4 Kbytes. The address of an entry in this table is VCT_BASE + VCOFFSET × 4 + VC pointer × 4.

The MPC8272 can check that all unallocated VCI bits are 0 by setting GMODE[CUAB] (check unallocated bits). If they are not, the cell is considered a misinserted cell.

An example of VC-level table entry address calculation is shown in Table 30-6. Note that VCOFFSET is assumed to be 0x100 for this example.

Table 30-6. VC-Level Table Entry Address Calculation Example

VCT_BASE	VCOffset	VC-Level Table Size	VC_MASK	VCI	VC Pointer	VC Entry Address
0x0084_0000	0x0100	32 entries	0x0037	0x0031	0x19	VC base = 0x840000 0x100 x 4 = 0x000400 0x19 x 4 = 0x000064 0x840464

Figure 30-8 shows the VC pointer address compression from Table 30-6.

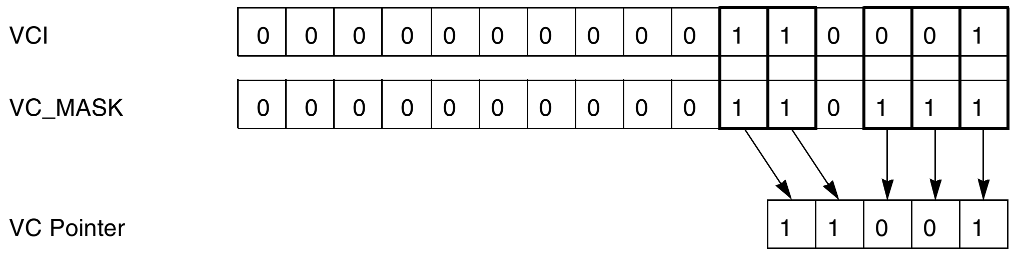


Figure 30-8. VC Pointer Address Compression

30.4.3 Misinserted Cells

If the address lookup mechanism cannot find a match (MS=1), the cell is discarded and ATM layer statistics are updated, as described in Section 30.8, “ATM Layer Statistics.”

30.4.4 Receive Raw Cell Queue

Channel one in the RCT is reserved as a raw cell queue. The user should program channel one to operate in AAL0 protocol. The receive raw cell queue is used for removing management cells from the regular cell flow to the host. When a management cell is sent to the receive raw cell queue, the CP sets RxBD[OAM]. The ALL0 BD specifies the channel code associated with the current OAM cell.

The following are optionally removed from the regular flow and sent to the raw cell queue:

- Segment F5 OAM (PTI = 0b100). To enable F5 segment filtering, set RCT[SEGF].
- End-to-end F5 OAM (PTI = 0b101). To enable F5 end-to-end filtering, set RCT[ENDF].
- RM cells (PTI = 0b110). When ABR flow is enabled the cells are terminated internally; otherwise, they are sent to the raw cell queue.
- Reserved PTI value (PTI = 0b111). Always sent to the raw cell queue.
- VCI value: 3, 4, 6, 7–15. To enable VCI filtering set the associated bit in the VCIF entry in the parameter RAM.

Figure 30-9 shows a flowchart of the ATM cell flow.

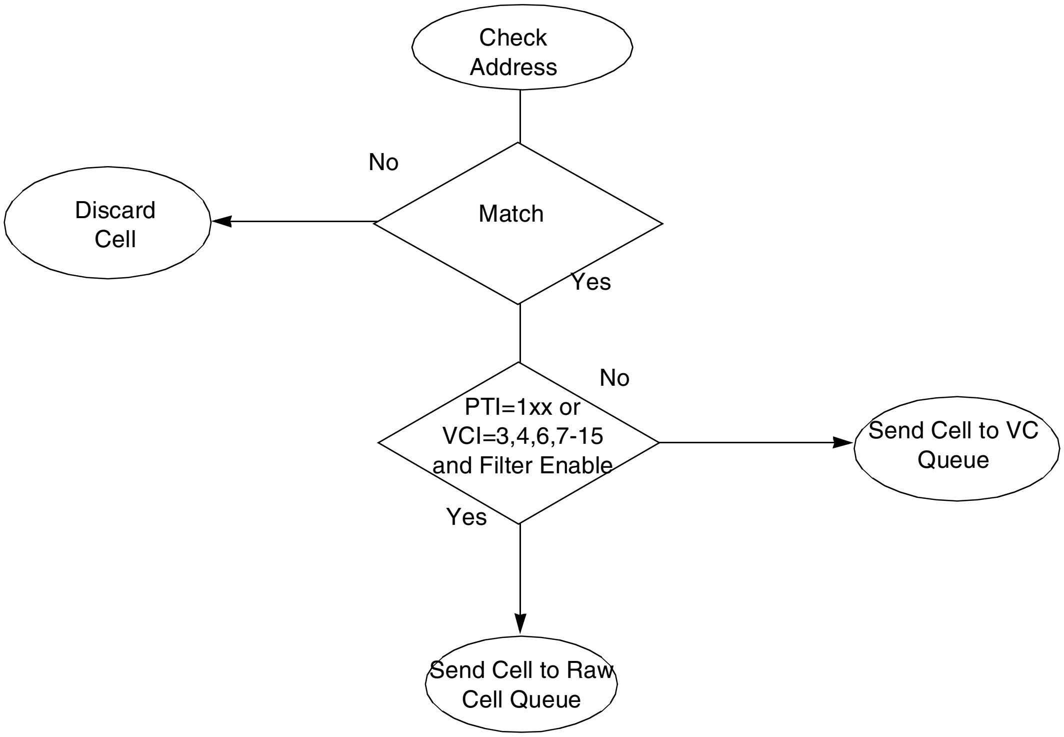


Figure 30-9. ATM Address Recognition Flowchart

NOTE

Even reserved VCI channels should appear in the CAM or address compression tables; otherwise, a cell on a reserved channel will be considered misinserted.

30.5 Available Bit Rate (ABR) Flow Control

While CBR service provides a fixed bandwidth and is useful for real-time applications with strictly bounded end-to-end cell transfer delay and cell-delay variation, ABR service is intended for data applications that can adapt to time-varying bandwidth and can tolerate significant cell transfer delay and cell delay variation. The MPC8272 implements the two following mechanisms defined by the ATM Forum TM 4.0 rate-based flow control.

- Explicit forward congestion indication (EFCI). The network supplies binary indication of whether congestion occurred along the connection path. This information is carried in the PTI field of the ATM cell header (similar to that used in frame relay). The source initially clears each ATM cell's EFCI bit, but as the cell passes through the connection, any congested node can set it. The MPC8272 detects this indication and sets the congestion indication (CI) bit in the next backwards RM cell to signal the source end station to reduce its transmission rate.
- Explicit rate (ER) feedback. The network carries explicit bandwidth information, to allow the source to adjust its rate. The source sends forward RM cells specifying its chosen transmit rate (source ER). A congested switch along the network may decrease ER to the exact rate it can support. The destination receives forward RM cells and returns them to the source as backward RM cells. The MPC8272 implements source behavior by adjusting the rate according to each returning backward RM cell's ER.

Explicit rate feedback has several advantages over binary feedback (EFCI). Explicit rate feedback allows immediate source rate adaptation, eliminating rate oscillation caused by incremental rate changes. Using the information in RM cells, the network can allocate bandwidth evenly among active ABR channels.

30.5.1 The ABR Model

Figure 30-10 shows the MPC8272's ABR model.

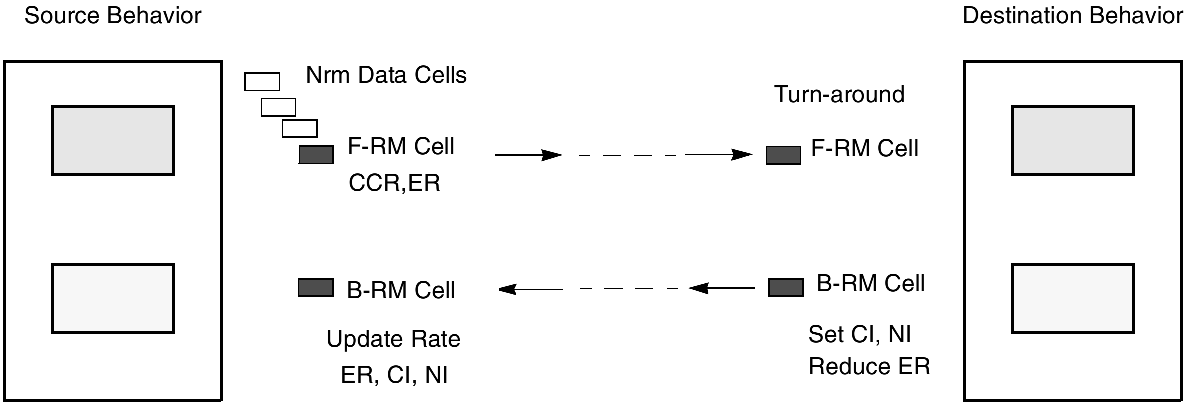


Figure 30-10. MPC8272's ABR Basic Model

The MPC8272 ABR flow control implements both source and destination behavior. The MPC8272's ABR flowchart is described in Section 30.5.1.3, "ABR Flowcharts."

30.5.1.1 ABR Flow Control Source End-System Behavior

The MPC8272’s implementation of ABR flow control for end-system sources is described in the following steps:

1. An ABR channel’s allowed cell rate (ACR) lies between the minimum cell rate (MCR) and the peak cell rate (PCR).
2. ACR is initialized to the initial cell rate (ICR).
3. An F-RM (Forward-RM) cell is sent for every Nrm data cell sent. If more than Mrm cells are sent and the time elapsed since the last F-RM exceeds Trm, an F-RM cell is sent.
4. When sending an F-RM cell, the current ACR is written in the CCR (current cell rate) field of the RM cell.
5. When B-RM (backward-RM) cell is received with CI = 1 (congestion indication), ACR is reduced by ACR × RDF (rate decrease factor). After the reduction, the new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).
6. When B-RM is received with CI=0 and NI=0 (no increase), ACR is increased by RIF × PCR (rate increase factor). The new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).
7. Before sending an F-RM cell, if more than ADTF (ACR decrease time factor) has elapsed since sending the last F-RM cell, ACR is reduced to ICR. In other words, if the source does not fully use its gained bandwidth, it loses it and resumes sending at its initial cell rate.
8. Before sending an F-RM cell and after action 7, if more than Crm F-RM cells were sent since the last B-RM cell was received with BN=0 (backward notification), the ACR is reduced by ACR × CDF (cut-off decrease factor).
9. A source whose ACR is less than the tag cell rate (TCR) sends out-of-rate cells at the TCR. This behavior is intended for sources whose rates were set to zero by the network. These sources should periodically sense the network state by sending out-of-rate RM cells. In this case data cells will not be sent.
10. An RM cell with an incorrect CRC10 is discarded and the UNI statistics tables are updated.

30.5.1.2 ABR Flow Control Destination End-System Behavior

The MPC8272’s implementation of ABR flow control for end-system destinations is described in the following steps:

1. A received F-RM cell is turned around and sent as a B-RM cells.
2. The DIR field of the received F-RM cell is changed from 0 to 1 (backward DIR).
3. The CCR and MCR fields are taken from the F-RM and is not changed.
4. The CI bit of the B-RM cell is set if the previous data cell arrived with EFCI = 1 (congestion bit in the ATM cell header).
5. The ER field of the turn around B-RM cells is limited by TCTE[ER-BRM].
6. If a F-RM cell arrives before the previous F-RM cell was turned around (for the same connection), the new RM cell overwrites the old RM cell.

30.5.1.3 ABR Flowcharts

The MPC8272's ABR transmit and receive flow control is described in the following flowcharts. See Figure 30-11, Figure 30-12, Figure 30-13, and Figure 30-14.

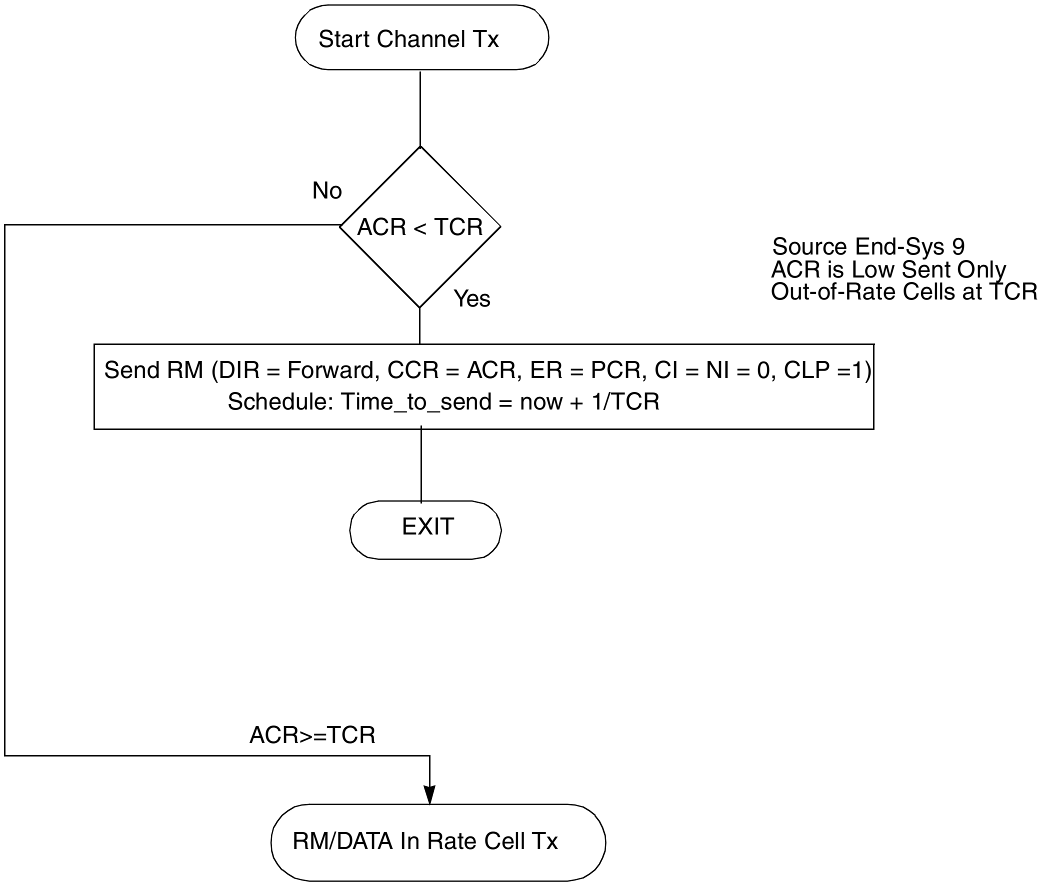


Figure 30-11. ABR Transmit Flow

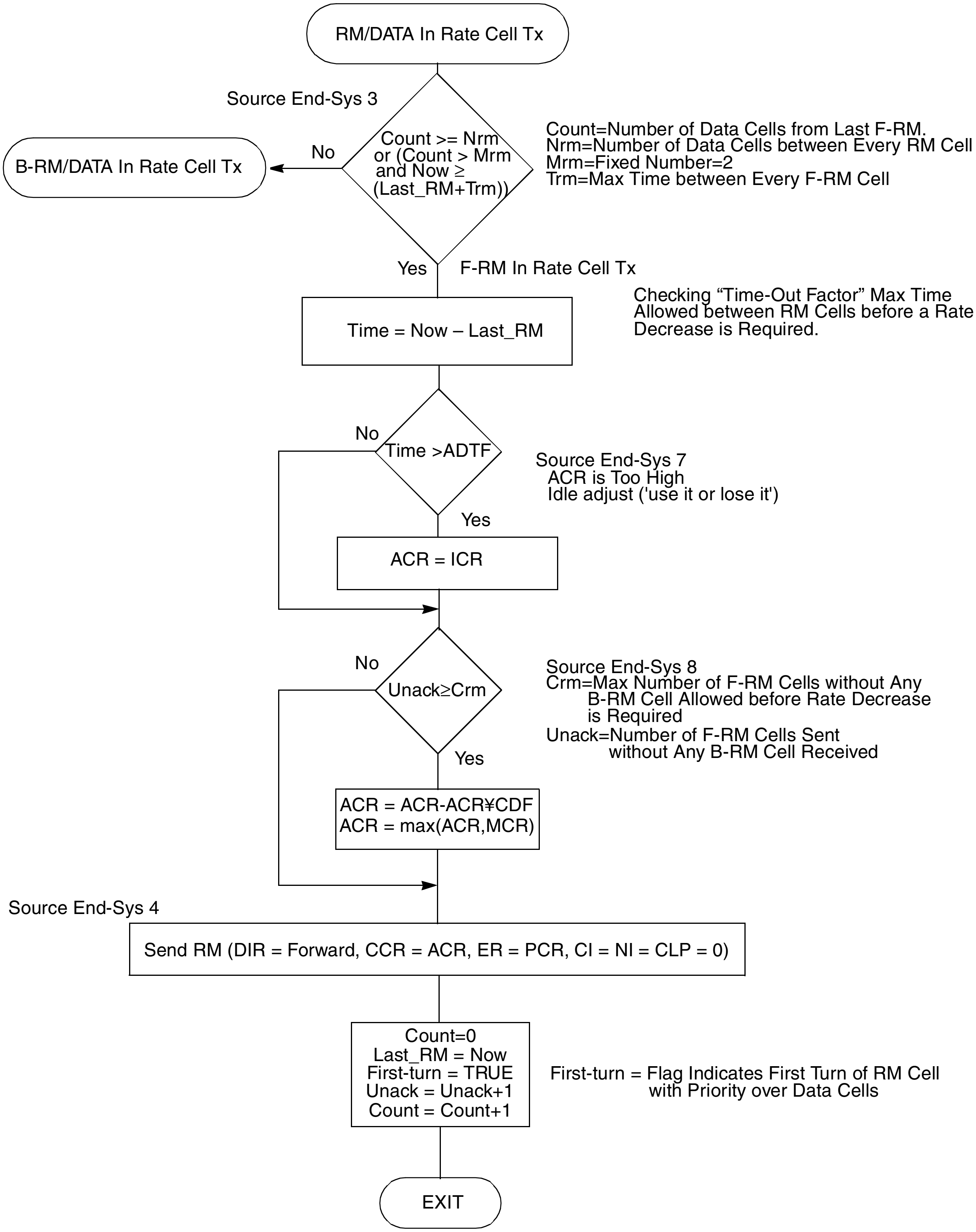

RM/DATA In Rate Cell Tx
Source End-Sys 3
Count >= Nrm or (Count > Mrm and Now ≥ (Last_RM+Trm))
No
B-RM/DATA In Rate Cell Tx
Count=Number of Data Cells from Last F-RM.
Nrm=Number of Data Cells between Every RM Cell
Mrm=Fixed Number=2
Trm=Max Time between Every F-RM Cell
Yes
F-RM In Rate Cell Tx
Time = Now – Last_RM
Checking “Time-Out Factor” Max Time Allowed between RM Cells before a Rate Decrease is Required.
No
Time >ADTF
Yes
Source End-Sys 7
ACR is Too High
Idle adjust ('use it or lose it')
ACR = ICR
No
Unack≥Crm
Yes
Source End-Sys 8
Crm=Max Number of F-RM Cells without Any B-RM Cell Allowed before Rate Decrease is Required
Unack=Number of F-RM Cells Sent without Any B-RM Cell Received
ACR = ACR-ACR¥CDF
ACR = max(ACR,MCR)
Source End-Sys 4
Send RM (DIR = Forward, CCR = ACR, ER = PCR, CI = NI = CLP = 0)
Count=0
Last_RM = Now
First-turn = TRUE
Unack = Unack+1
Count = Count+1
First-turn = Flag Indicates First Turn of RM Cell with Priority over Data Cells
EXIT

Figure 30-12. ABR Transmit Flow (Continued)

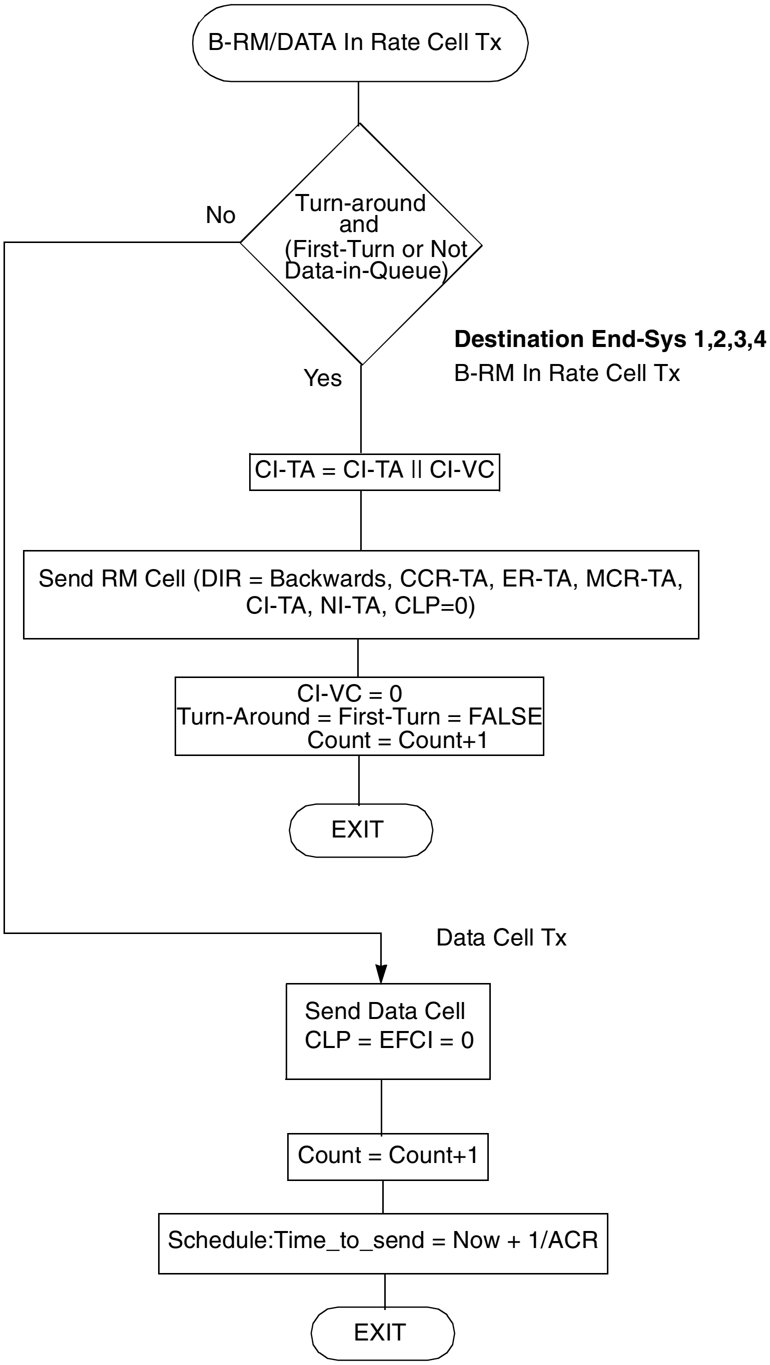

B-RM/DATA In Rate Cell Tx
Turn-around and (First-Turn or Not Data-in-Queue)
No
Yes
Destination End-Sys 1,2,3,4
B-RM In Rate Cell Tx
CI-TA = CI-TA II CI-VC
Send RM Cell (DIR = Backwards, CCR-TA, ER-TA, MCR-TA, CI-TA, NI-TA, CLP=0)
CI-VC = 0
Turn-Around = First-Turn = FALSE
Count = Count+1
EXIT
Data Cell Tx
Send Data Cell
CLP = EFCI = 0
Count = Count+1
Schedule:Time_to_send = Now + 1/ACR
EXIT

Figure 30-13. ABR Transmit Flow (Continued)

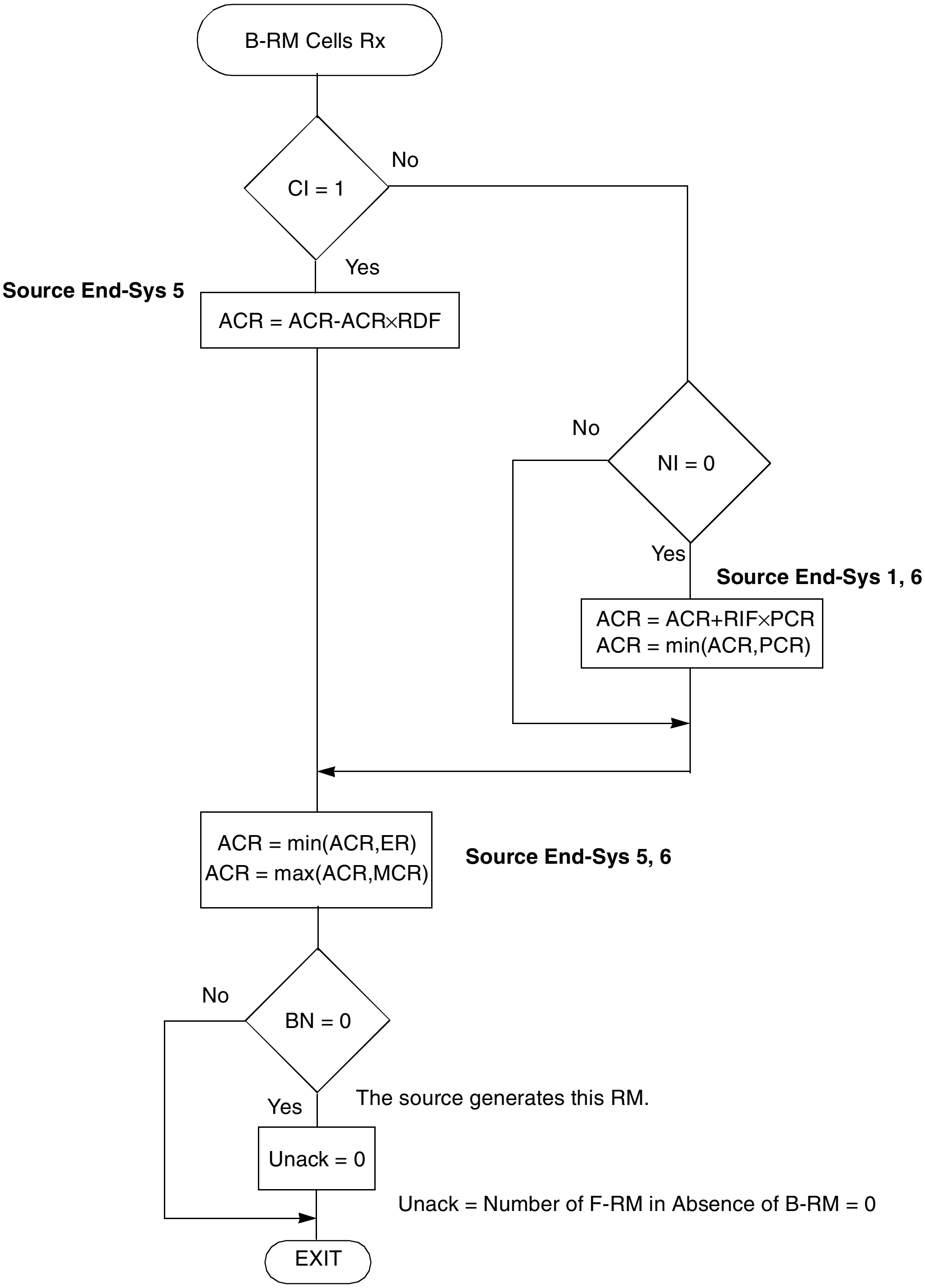


Figure 30-14. ABR Receive Flow

30.5.2 RM Cell Structure

Table 30-7 describes the structure of the RM cell supported by the MPC8272. For more information, see the ABR flow-control traffic management specification (TM 4.0) on the ATM Forum website.

Table 30-7. Fields and their Positions in RM Cells

Fields	Octet	Bits	Description	Value
Header	1–5	All	ATM cell header	RM-VCC PTI=6
ID	6	All	Protocol ID	1
DIR	7	0	Direction of RM cell (0 = forward, 1 = backward)	
BN	7	1	Backward notification (BN = 0, the cell was generated by the source; BN=1, the cell was generated by the network or by the destination)	
CI	7	2	Congestion indication. (1 = congestion, 0 = otherwise)	
NI	7	3	No increase indication. (1 = no increase allowed, 0 = otherwise)	
RA	7	4	Not used (ATM Forum ABR)	0
—	7	5-7	Reserved, should be cleared.	0
ER	8–9	All	Explicit rate; see Section 30.5.2.1.	
CCR	10–11	All	Current cell rate; see Section 30.5.2.1.	
MCR	12–13	All	Minimum cell rate; see Section 30.5.2.1.	
QL	14–17	All	Not used (ATM Forum ABR)	0
SN	18–21	All	Not used (ATM Forum ABR)	0
—	22–51	All	Reserved, should be cleared.	0x6A for each byte
—	52	0–5	Reserved, should be cleared.	0
CRC-10	52	6–7	CRC-10	
	53	All		

30.5.2.1 RM Cell Rate Representation

Rates in the RM cells are represented in a binary floating-point format using a 5-bit exponent (e), a 9-bit mantissa (m), and a 1-bit nonzero flag (nz), as shown in Figure 30-15.

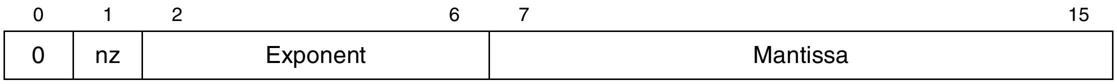


Figure 30-15. Rate Format for RM Cells

The rate (in cells/second) is calculated as in Figure 30-16.

$$\text{Rate} = \left[2^{e} \times \left(1 + \frac{m}{512}\right)\right] \times nz$$

Figure 30-16. Rate Formula for RM Cells

Initialize the traffic parameters (ER, MCR, PCR, or ICR) in the ABR protocol-specific connection tables using the rate formula in Figure 30-16.

30.5.3 ABR Flow Control Setup

Follow these steps to setup ABR flow control:

1. Initialize the ABR data structure: RCT, TCT, RCT-ABR protocol-specific, TCTE-ABR protocol-specific.
2. Initialize ABR global parameters in the parameter RAM. See Section 30.10.1, "Parameter RAM."
3. Program the AAL-type in the RCT and TCT to AAL5 and set TCT[ABRF].

NOTE

ABR flow control is available only with AAL5.

4. The time stamp timer generates the RM cell's time stamp, which the ABR flow control monitors to maintain source behavior in steps #3 and #7 of Section 30.5.1.1, "ABR Flow Control Source End-System Behavior." Enable the time stamp timer by writing to the RTSCR; see Section 13.3.7, "RISC Time-Stamp Control Register (RTSCR)."
5. Initialize the ABR parameters (CPS_ABR and LINE_RATE_ABR) in the APCT; see Section 30.10.4.1, "APC Parameter Tables." Note that when using ABR, the CPS (cells per slot) parameter in the APCPT should be a power of two.
6. Finally, send the ATM TRANSMIT command to restart channel transmission.

30.6 OAM Support

This section describes the MPC8272's support for ATM-layer (F4 out-of-band, and F5 in-band) operations and maintenance (OAM) of connections. Alarm surveillance, continuity checking, remote defect indication, and loopback cells are supported using OAM receive and transmit AAL0 cell queues. Using dedicated support, performance management block tests can be performed on up to 64 connections simultaneously. The CP automatically inserts forward monitoring cells (FMC) and generates backward-reporting cells (BRC) as recommended by ITU I.610.

30.6.1 ATM-Layer OAM Definitions

Table 30-8 lists pre-assigned header values at the user-network interface (UNI).

Table 30-8. Pre-Assigned Header Values at the UNI

Use	GFC	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

Note: a = available for use by the appropriate ATM layer function.

Table 30-9 lists pre-assigned header values at the network-node interface (NNI).

Table 30-9. Pre-Assigned Header Values at the NNI

Use	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

Note: a= available for use by the appropriate ATM layer function.

30.6.2 Virtual Path (F4) Flow Mechanism

The F4 flow is designated by pre-assigned virtual channel identifiers within the virtual path. The following two kinds of F4 flows can exist simultaneously:

- End-to-end (identified as VCI 4)—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)—Used for communicating operations information within one VPC link or among multiple interconnected VPC links. The concatenation of VPC links is called a VPC segment. Cells inserted into this flow can be removed only by the segment endpoints, which must remove these cells to prevent confusion in adjacent segments.

30.6.3 Virtual Channel (F5) Flow Mechanism

The F5 flow is designated by pre-assigned payload type identifiers. The following two kinds of F5 flow can exist simultaneously:

- End-to-end (identified by PTI = 5)—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)—Used for communicating operations information with the bound of one VCC link or multiple interconnected VCC links. A concatenation of VCC links is called a VCC segment. Segment endpoints must remove these cells to prevent confusion in adjacent segments.

30.6.4 Receiving OAM F4 or F5 Cells

OAM F4/F5 flow cells are received using the raw cell queue, described in Section 30.4.4, “Receive Raw Cell Queue.” An F4/F5 OAM cell which does not appear in the CAM or address compression tables is considered a misinserted cell.

30.6.5 Transmitting OAM F4 or F5 Cells

OAM F4/F5 flow cells are sent using the usual AAL0 transmit flow. For OAM F4/F5 cell transmission, program channel one in the TCT to operate in AAL0 mode. Enable the CR10 (CRC-10 insertion) mode as described in Section 30.10.2.3.3, “AAL0 Protocol-Specific TCT.” Prepare the OAM F4/F5 flow cell and insert it in an AAL0 TxBD. Finally, issue a ATM TRANSMIT command to send the OAM cell. For multiple

PHYs, use several AAL0 channels—each PHY should have one transmit raw cell queue that is associated with its scheduling table.

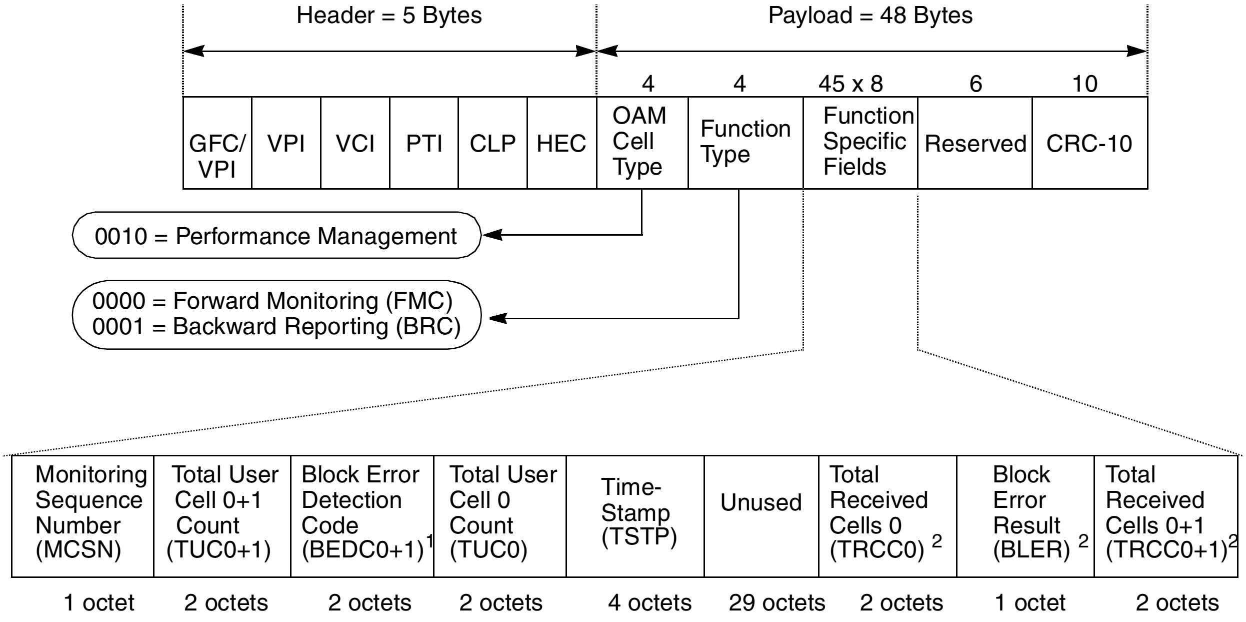
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel's TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are processed).

30.6.6 Performance Monitoring

A connection's performance is monitored by inspecting blocks of cells (delimited by forward monitoring cells) sent between connection or segment endpoints. Each FMC contains statistics about the immediately preceding block of cells. When an endpoint receives an FMC, it adds the statistics generated locally across the same block to produce a backward reporting cell (BRC), which is then returned to the opposite endpoint.

The MPC8272 can run up to 64 bidirectional block tests simultaneously. When a bidirectional test is run, FMCs are generated for one direction and checked for the opposite.

Figure 30-17 shows the FMC and BRC cell structure.


Header = 5 Bytes
Payload = 48 Bytes
4
4
45 x 8
6
10
GFC/ VPI
VPI
VCI
PTI
CLP
HEC
OAM Cell Type
Function Type
Function Specific Fields
Reserved
CRC-10
0010 = Performance Management
0000 = Forward Monitoring (FMC)
0001 = Backward Reporting (BRC)
Monitoring Sequence Number (MCSN)
Total User Cell 0+1 Count (TUC0+1)
Block Error Detection Code (BEDC0+1)1
Total User Cell 0 Count (TUC0)
Time-Stamp (TSTP)
Unused
Total Received Cells 0 (TRCC0) 2
Block Error Result (BLER) 2
Total Received Cells 0+1 (TRCC0+1)2
1 octet
2 octets
2 octets
2 octets
4 octets
29 octets
2 octets
1 octet
2 octets

[1] $BEDC_{0+1}$ appears in FMCs only.

[2] $TRCC_0$, BLER, and $TRCC_{0+1}$ appear in BRCs only.

Figure 30-17. Performance Monitoring Cell Structure (FMCs and BRCs)

Table 30-10 describes performance monitoring cell fields.

Table 30-10. Performance Monitoring Cell Fields

Field	Description	BRC	FMC
MCSN	Monitoring cell sequence number. The sequence number of the performance monitoring cell (modulo 256).	Yes	Yes
TUC_{0+1}	Total user cell 0+1 count. Counts all user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TUC_0	Total user cell 0 count. Counts CLP = 0 user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TSTP	Time stamp. Used to indicate when the cell was inserted.	Yes	Yes
$BEDC_{0+1}$	Block error detection code. Even parity over the payload of the block of user cells sent since the last FMC.	No	Yes
$TRCC_0$	Total received cell count 0. Counts CLP=0 user cells (modulo 65,536) received before the FMC was received.	Yes	No
BLER	Block error result. Counts error parity bits detected by the BEDC of the received FMC.	Yes	No
$TRCC_{0+1}$	Total received cell count 0+1. Counts all user cells (modulo 65,536) received before the FMC was received.	Yes	No

30.6.6.1 Running a Performance Block Test

For bidirectional PM block tests, FMCs are monitored at the receive side and generated at the transmit side. The following setup is required to run a bidirectional PM block test on an active VCC:

1. Assign one of the available 64 performance monitoring tables by writing to both RCT[PMT] and TCT[PMT] and initializing the one chosen. See Section 30.10.3, “OAM Performance Monitoring Tables.”
2. For PM F5 segment termination set RCT[SEGF]; for PM F5 end-to-end termination set RCT[ENDF].
3. Finally, set the channel’s RCT[PM] and TCT[PM] and the receive raw cell’s RCT[PM].

For unidirectional PM block tests:

- For PM block monitoring only, set only the RCT fields above.
- For PM block generation only, set only the TCT fields above.

To run a block test on a VPC, assign all the VCCs of the tested VPC to the same performance monitoring table. Configure RCT[PMT] and TCT[PMT] to specify the performance monitoring table associated with each F4 channel.

30.6.6.2 PM Block Monitoring

PM block monitoring is done by the receiver. After initialization (see Section 30.6.6.1), whenever a cell is received for a VCC or VPC, the TRCC counters are incremented and the BEDC is calculated. When an FMC is received, the CP adds the BRC fields into the cell payload ($TRCC_0$, $TRCC_{0+1}$, BLER) and transfers the cell to the receive raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

Before the BRC is transferred to the transmit raw cell queue, the PM function type should be changed to backward reporting and additional checking should be done regarding the BLER field. If the sequence numbers (MCSN) of the last two FMCs are not sequential or the differences between the last two TUCs and the last two TRCCs are not equal, BLER should be set to all ones (see the ITU I.610 recommendation).

NOTE

> TRCCs are free-running counters (modulo 65,536) that count user cells received. The total received cells of a particular block is the difference between TRCC values of two consecutive BRC cells. TRCC values are taken from a VC’s performance monitoring table.

30.6.6.3 PM Block Generation

The transmitter generates the PM block. Each time the transmitted cell count parameter (TCC) in the performance monitoring table reaches zero, the CP inserts an FMC into the user cell stream. The CP copies the FMC header, SN-FMC, TUC_{0+1}, TUC_0, $BEDC_{0+1}$-Tx from the performance monitoring table and inserts them into the FMC payload. The TSTP value (FMC time stamp field) is taken from the MPC8272 time stamp timer; see Section 13.3.7, “RISC Time-Stamp Control Register (RTSCR).”

The TUCs are free-running counters (modulo 65,536) that count transmitted user cells. The total transmitted cells of a particular block is the difference between TUC values of two consecutive FMCs. The BEDC (BIP-16, bit interleaved parity) calculation is done on the payload of all user cells of the current tested block. The performance monitoring block can range from 1 to 2K cells, as specified in the BLCKSIZE parameter in the performance monitoring table; see Section 30.10.3, “OAM Performance Monitoring Tables.”

In Figure 30-18, the performance monitoring block size is 512 cells. For every 512 user cells sent, the ATM controller automatically inserts an FMC into the regular cell stream as defined in ITU I.610. When an FMC is received, the ATM controller adds the BRC fields to the cell payload and sends the cell to the raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

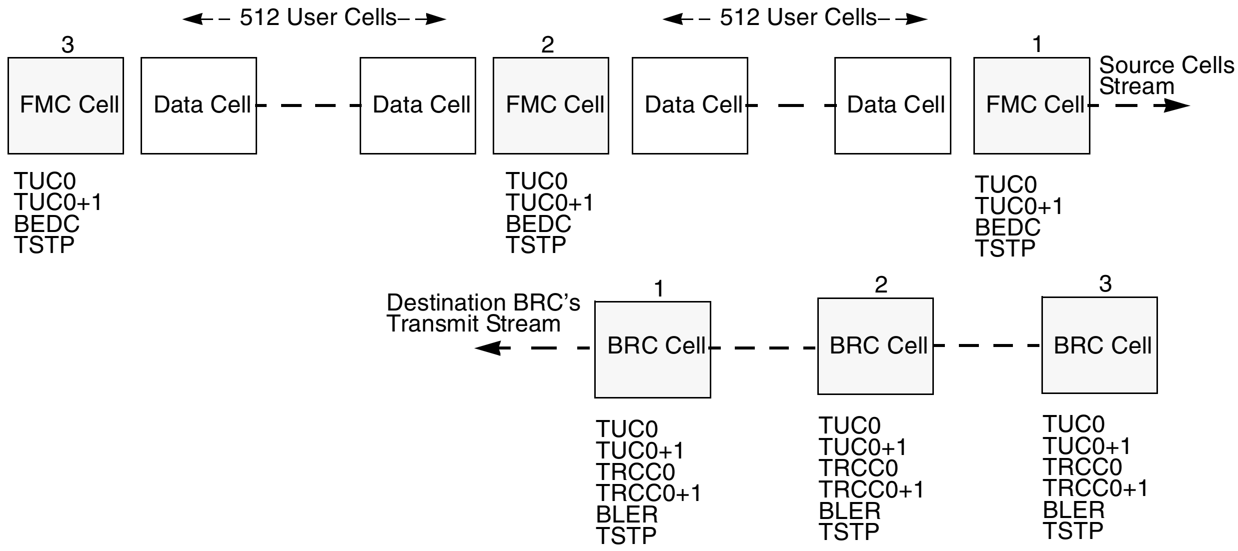


Figure 30-18. FMC, BRC Insertion

30.6.6.4 BRC Performance Calculations

BRC reception uses the regular AAL0 raw cell queue. On receiving two consecutive BRC cells, the management layer can calculate the following:

- The difference between two TUCs (Nt)
- The difference between two TRCCs (Nr)

Information about the connection can be gained by comparing Nt and Nr:

- If Nt > Nr, the difference indicates the number of lost cells of this block test.
- If Nt < Nr, the difference indicates the number of misinserted cells of this block test.
- When Nt = Nr, no cells are lost or misinserted.

30.7 User-Defined Cells (UDC)

Typical ATM cells are 53 bytes long and consist of a 4-byte header, 1-byte HEC, and 48-byte payload. The MPC8272 also supports user-defined cells with up to 12 bytes of extra header fields for internal information for switching applications. This choice is made during initialization by writing to the FPSMR; see Section 30.13.3, “FCC Protocol-Specific Mode Register (FPSMR).” As shown in Figure 30-19, the extra header size can vary between 1 to 12 bytes (byte resolution) and the HEC octet is optional.

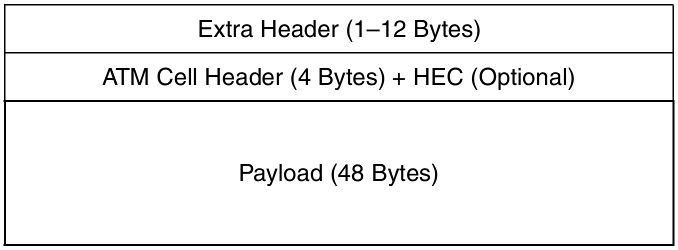

Extra Header (1–12 Bytes)
ATM Cell Header (4 Bytes) + HEC (Optional)
Payload (48 Bytes)

Figure 30-19. Format of User-Defined Cells

For AAL5 the extra header is taken from the Rx and Tx BDs. The transmitter reads the extra header from the UDC TxBD and adds it to each ATM cell associated with the current buffer. At the receive side, the extra header of the last cell in the current buffer is written to the UDC RxBD.

For AAL0 the extra header is attached to the regular ATM cell in the buffer. The transmitter reads the extra header and the ATM cell from the buffer. The receiver writes the extra header and the regular ATM cell to the buffer.

30.7.1 UDC Extended Address Mode (UEAD)

For external CAM accesses, the UDC extra header can be used to supply extra routing information; see Figure 30-20. If GMODE[UEAD] = 1, 2 bytes of the UDC header are used as extensions to the ATM address and the CAM match cycle performs a double-word access. UEAD_OFFSET in the parameter RAM determines the offset from the beginning of the UDC extra header to the UEAD entry. The offset should be half-word aligned (even address). See Section 30.10.1, “Parameter RAM.”

	16-Bit	4-Bit	12-Bit	16-Bit
CAM Data in Field:	UEAD	PHY Addr	VPI	VCI
	0 15	16 19	20 31	32 47

Figure 30-20. External CAM Address in UDC Extended Address Mode

30.8 ATM Layer Statistics

ATM layer statistics can be used to identify problems, such as the line-bit error rate, that affect the UNI performance. Statistics are kept in three 16-bit wrap-around counters:

- UTOPIA error dropped cells count—Counts cells discarded due to UTOPIA errors: Rx parity errors and short or long cells
- Misinserted dropped cell count—Counts cells discarded due to address look-up failure
- CRC10 error dropped cell count—Counts cells discarded due to CRC10 errors. (ABR only)

Counters are implemented in the dual-port RAM for each PHY device. The counters of each PHY are located in the UNI statistics table, described in Section 30.10.7, “UNI Statistics Table.”

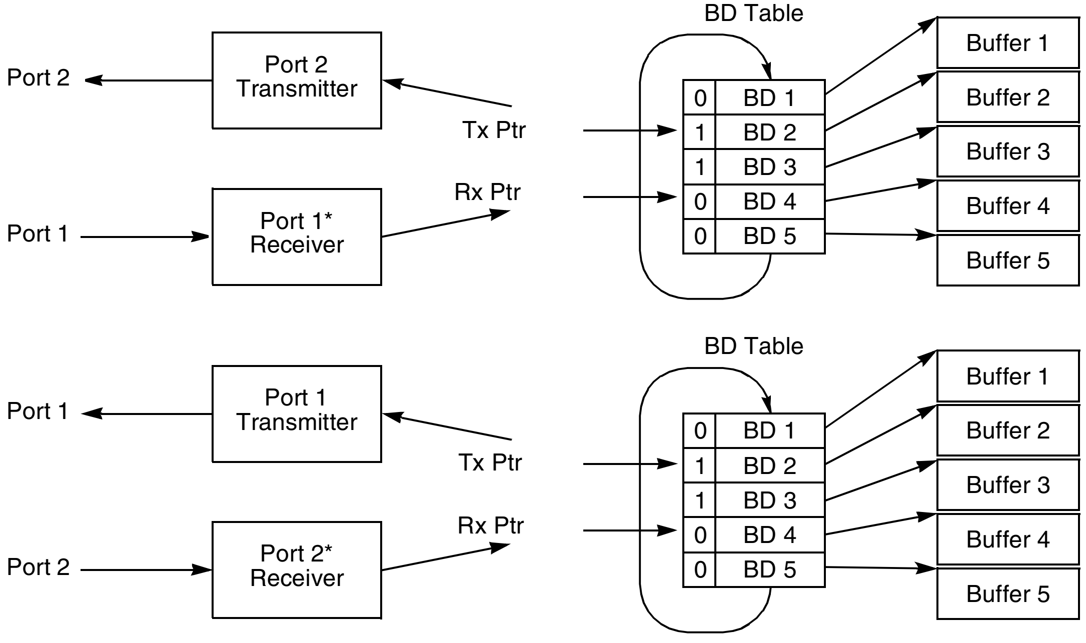
30.9 Interworking

The MPC8272 supports port-to-port interworking. Data forwarding between two ATM ports can be done in two ways:

- Core intervention—When a receive buffer in one ATM port is full and its RxBD is closed, that port interrupts the core. The core copies the port’s receive buffer pointer to the second ATM port’s TxBD and sets the ready bit (TxBD[R]). This mode is useful when additional core processing is required.
- Automatic data forwarding. This mode enables automatic data forwarding between AAL1/AAL0 and transparent mode over an ATM port. See Section 30.9.1, “ATM-to-ATM Automatic Data Forwarding.”

30.9.1 ATM-to-ATM Automatic Data Forwarding

Automatic data forwarding can be used to switch ATM AAL0 cells from one ATM port to another without core intervention. The receiver of one port and transmitter of the other port should be programed to process the same BD table. When the one port’s receiver fills an AAL0 buffer, the other port’s transmitter sends it, as shown in Figure 30-21.



* The receivers should be programmed to operate in opposite polarity E (empty) bit.

Figure 30-21. ATM-to-ATM Data Forwarding

As Figure 30-21 shows, when data is being forwarded from ATM port 2 to ATM port 1, the receiver of port 2 reassembles data received from a particular channel to a specific BD table. The transmitter of port 1 is programmed to operate on the same table. When port 2 fills a receive buffer, the port transmits it. The two ATM ports synchronize on port 2's RxBD[E] and port 1's TxBD[R].

When data is being forwarded from port 1 to port 2, the receiver of port 1 routes data to a specific BD table and port 2's transmitter is programmed to operate on the same table. When port 1 fills a receive buffer, the port 2 sends it. The controllers synchronize on port 1's RxBD[E] and the port 2's TxBD[R].

The receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive data into buffers whose RxBD[E] = 0 and set RxBD[E] when a buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1- and AAL0-specific areas of the receive connection table.; see Section 30.10.2.2, "Receive Connection Table (RCT)."

30.9.2 Using Interrupts in Automatic Data Forwarding

The core can program the interrupt mechanism of the ATM to trigger interrupts for events such as a buffer closing or transfer errors. The interrupt mechanism can be used to synchronize the start of the automatic bridging process. For example, to start the transmitter of a one ATM port after a specific buffer reaches the other ATM port's receiver (the buffering is required to cope with the ATM network's CDV), set RxBD[I] in the second ATM port. When the receive buffer is full, the RxBD is closed, RxBD[E] is set (because it is operating in opposite E-bit polarity), and the core is interrupted. The core then starts the first port's transmitter.

30.10 ATM Memory Structure

The ATM memory structure, described in the following sections, includes the parameter RAM, the connection tables, OAM performance monitoring tables, the APC data structure, BD tables, and the UNI statistics table.

30.10.1 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 30-11. Note that there are additional parameters for AAL2 (refer to Table 31-13).

Table 30-11. ATM Parameter RAM Map

Offset[1]	Name	Width	Description
0x00–0x3F	—	—	Reserved, should be cleared.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User-defined offset from dual-port RAM base.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined offset from dual-port RAM base.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined offset from dual-port RAM base.
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined offset from dual-port RAM base.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined offset from dual-port RAM base.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined offset from dual-port RAM base.
0x4C	—	Word	Reserved, should be cleared.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. Offset to the user-defined extended address (UEAD) in the UDC extra header. Must be an even address. See Section 30.10.1.1, “Determining UEAD_OFFSET (UEAD Mode Only).” If RCT[BO] = 01, UEAD_OFFSET should be in little-endian format. For example, if the UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be programmed to 2 (second half word entry in dual-port RAM).
0x5E	—	Hword	Reserved, should be cleared.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined offset from dual-port RAM base.

Table 30-11. ATM Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0x62	APCP_BASE	Hword	APC parameter table base address. User-defined offset from dual-port RAM base.
0x64	FBT_BASE	Hword	Free buffer pool parameter table base. User-defined offset from dual-port RAM base.
0x66	INTT_BASE	Hword	Interrupt queue parameter table base. User-defined offset from dual-port RAM base.
0x68	—	—	Reserved, should be cleared.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined offset from dual-port RAM base. Note that this must be set up according to Section 30.10.7, “UNI Statistics Table.” It is not optional.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] holds the 8 most-significant bits of the Rx/Tx BD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCIF	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7-15 are received and the associated VCIF bit = 1 the cell is sent to the raw cell queue. VCIF[0–2, 5] should be zero. See Section 30.10.1.2, “VCI Filtering (VCIF).”
0x84	GMODE	Hword	Global mode. User-defined. See Section 30.10.1.3, “Global Mode Entry (GMODE).”
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 30.15, “ATM Transmit Command.”
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved, should be cleared.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFF_FFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB_20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SNP protection look-up table base address. (AAL1 CES only.) The 32-byte table resides in dual-port RAM. AAL1_SNPT_BASE must be half word–aligned. User-defined offset from dual-port RAM base. See Section 30.10.6, “AAL1 Sequence Number (SN) Protection Table.”
0x9A	—	Hword	Reserved, should be cleared.
0x9C	SRTS_BASE	Word	External SRTS logic base address. AAL1 CES only. Should be 16-byte aligned. The four least-significant bits are taken from SRTS_DEV in the AAL1-specific area of the connection table entries.

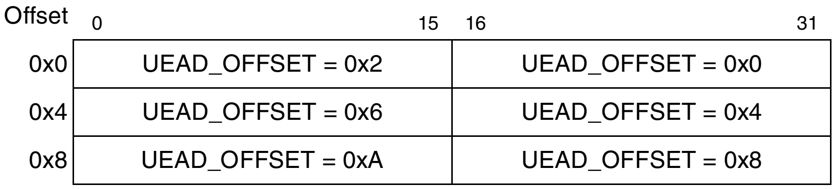
Table 30-11. ATM Parameter RAM Map (continued)

Offset[1]	Name	Width	Description
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined offset from dual-port RAM base. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/unassign cell size. 52 in regular mode; 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A_6A6A.
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler; see Section 13.3.7, “RISC Time-Stamp Control Register (RTSCR).” For time stamp prescalar of 1 μs, program Trm to be 100 ms/1μs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate (MCR) should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined offset from dual-port RAM base.
—	Additional parameters	—	• For additional AAL1 CES parameters, refer to Table 31-4. • For additional AAL2 parameters, refer to Table 31-13.

[1] Offset from FCC base: 0x8400 (FCC1); see Section 13.5.2, “Parameter RAM.”

30.10.1.1 Determining UEAD_OFFSET (UEAD Mode Only)

The UEAD_OFFSET value is based on the position of the user-defined extended address (UEAD) in the UDC extra header. Figure 30-22 shows how to determine UEAD_OFFSET: first determine the half word–aligned location of the UEAD and then read the corresponding UEAD_OFFSET value.



Offset	0 — 15	16 — 31
0x0	UEAD_OFFSET = 0x2	UEAD_OFFSET = 0x0
0x4	UEAD_OFFSET = 0x6	UEAD_OFFSET = 0x4
0x8	UEAD_OFFSET = 0xA	UEAD_OFFSET = 0x8

Figure 30-22. UEAD_OFFSETs for Extended Addresses in the UDC Extra Header

30.10.1.2 VCI Filtering (VCIF)

VCI filtering enable bits are shown in Figure 30-23.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	0	VC3	VC4	0	VC6	VC7	VC8	VC9	VC10	VC11	VC12	VC13	VC14	VC15

Figure 30-23. VCI Filtering Enable Bits

Table 30-12 describes the operation of the VCI filtering enable bits.

Table 30-12. VCI Filtering Enable Field Descriptions

Bits	Name	Description
0–2, 5	—	Clear these bits.
3, 4, 6, 7–15	VC*x*	VCI filtering enable 0 Do not send cells with this VCI to the raw cell queue. 1 Send cells with this VCI to the raw cell queue.

30.10.1.3 Global Mode Entry (GMODE)

Figure 30-24 shows the layout of the global mode entry (GMODE).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	GBL	0	0	0	ALB	CTB	REM	0	0	UEAD	CUAB	EVPT	0	ALM

Figure 30-24. Global Mode Entry (GMODE)

Table 30-13 describes GMODE fields.

Table 30-13. GMODE Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
3–5	—	Reserved, should be cleared.
6	ALB	Address look up bus for CAM or address compression tables 0 Reside on the 60x bus 1 Reserved
7	CTB	External connection tables bus 0 Reside on the 60x bus 1 Reserved
8	REM	Receive emergency mode 0 Enable REM operation. When the receive FIFO is full, the ATM transmitter stops sending data cells until the receiver emergency state is cleared (FIFO not full). The transmitter pace is maintained, although a small CDV may be introduced. This mode enables the receiver to receive bursts of cells above the steady state performance. 1 Disable REM operation. Note that to check system performance the user may want to set this bit.
9–10	—	Reserved, should be cleared.

Table 30-13. GMODE Field Descriptions (continued)

Bits	Name	Description
11	UEAD	User-defined cells extended address mode. See Section 30.7.1, “UDC Extended Address Mode (UEAD).” 0 Disable UEAD mode. 1 Enable UEAD mode.
12	CUAB	Check unallocated bits 0 Do not check unallocated bits during address compression. 1 Check unallocated bits during address compression.
13	EVPT	External address compression VP table 0 VP table resides in dual-port RAM 1 VP table reside in external memory
14	—	Reserved, should be cleared.
15	ALM	Address look-up mechanism. See Section 30.4, “VCI/VPI Address Lookup Mechanism.” 0 External CAM lookup 1 Address compression

30.10.2 Connection Tables (RCT, TCT, and TCTE)

The receive and transmit connection tables, RCT and TCT, store host-initialized connection parameters after connection set-up. These include AAL type, connection traffic parameters, BD parameters and temporary parameters used during segmentation and reassembly (SAR). The transmit connection table extension (TCTE) supports special connections that use ABR, VBR or UBR+ services. Each connection table entry resides in a 32-byte space. Table 30-14 lists sizes for RCT, TCT, and TCTE.

Table 30-14. Receive and Transmit Connection Table Sizes

ATM Service Class	RCT	TCT	TCTE
CBR, UBR service	32 bytes	32 bytes	—
ABR, VBR, UBR+ service	32 bytes	32 bytes	32 bytes

NOTE

An ATM channel is considered internal if its tables are in an internal dual-port RAM; it is considered external if its tables are in external memory. To improve performance, store parameters for fast channels in internal dual-port RAM and parameters for slower channels in external memory. Connection tables for external channels are read and written from external memory each time the CP processes a cell. The CP does, however, minimize memory access time by burst fetching the 32-byte entry and writing back only the first 24 bytes.

In all connection tables, fields that are not used must be cleared.

30.10.2.1 ATM Channel Code

Each ATM channel has a channel code used as an index to the channel's connection table entry. The first channel in the table has channel code one, the second has channel code two, and so on. Codes of 255 or less indicate internal channels; codes greater than 255 indicate external channels. Channel code one is reserved as the raw cell queue and cannot be used for another purpose. The channel code is used to specify a VC when sending a ATM TRANSMIT command, initiating the external CAM or address compression tables, and when the CP sends an interrupt to an interrupt queue.

Example:

Suppose a configuration supports 1,024 regular ATM channels. To allocate 4 Kbytes of dual-port RAM space to the internal connection table, determine that channel codes 0–63 are internal (64 VCs × 64 bytes (RCT and TCT) = 4 K). Channels 0–1 are reserved. The remaining 962 (1024 – 62) external channels are assigned channel codes 256–1217. See Figure 30-25.

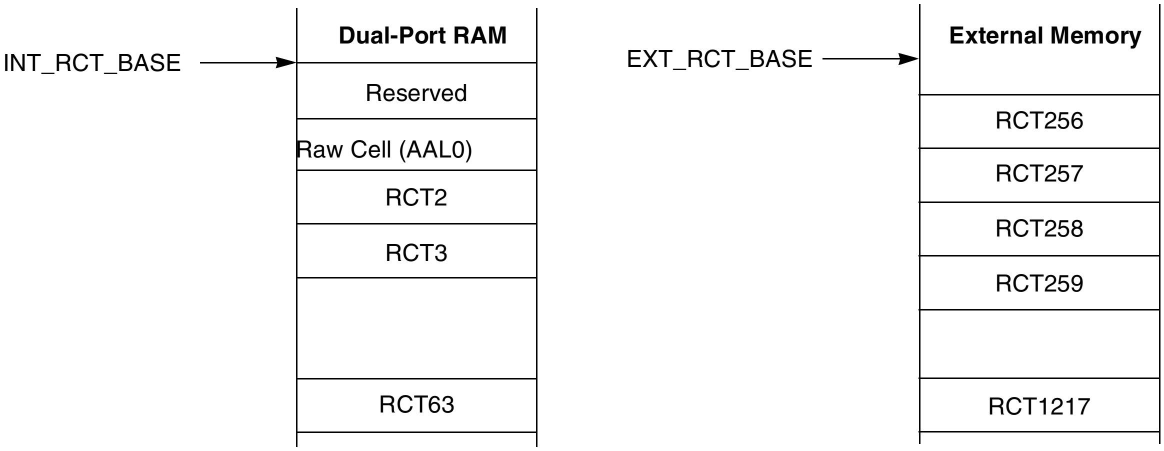


Figure 30-25. Example of a 1024-Entry Receive Connection Table

The general formula for determining the real starting address for all internal and external connection table entries is as follows:

Connection table base address + (channel code × 32)

Thus, the real starting address of the RCT entry associated with channel code 3 is as follows:

INT_RCT_BASE+ (3 × 32) = INT_RCT_BASE + 96

Even though it produces a gap in the connection table, the first external channel's real starting address of the RCT entry (channel code 256) is as follows:

EXT_RCT_BASE+ (256 × 32) = EXT_RCT_BASE + 8192

See Section 30.10.1, "Parameter RAM," to find all the connection table base address parameters. (The transmit connection table base address parameters are INT_TCT_BASE, EXT_TCT_BASE, INT_TCTE_BASE, and EXT_TCTE_BASE.)

30.10.2.2 Receive Connection Table (RCT)

Figure 30-26 shows the format of an RCT entry.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x00</td><td colspan="2">—</td><td>GBL</td><td colspan="2">BO</td><td>—</td><td>DTB</td><td>BIB</td><td>—</td><td>BUFM</td><td>SEGF</td><td>ENDF</td><td colspan="2">—</td><td colspan="2">INTQ</td></tr>
<tr><td>Offset + 0x02</td><td>—</td><td>INF</td><td colspan="10">—</td><td>ABRF</td><td colspan="3">AAL</td></tr>
<tr><td>Offset + 0x04</td><td colspan="16" rowspan="2">RX Data Buffer Pointer (RXDBPTR)</td></tr>
<tr><td>Offset + 0x06</td></tr>
<tr><td>Offset + 0x08</td><td colspan="16" rowspan="2">Cell Time Stamp</td></tr>
<tr><td>Offset + 0x0A</td></tr>
<tr><td>Offset + 0x0C</td><td colspan="16">RBD_Offset</td></tr>
<tr><td>Offset + 0x0E</td><td colspan="16" rowspan="6">Protocol Specific
• For AAL5,Section 30.10.2.2.1, “AAL5 Protocol-Specific RCT.”
• For AAL2, Section 31.4.4.1, “AAL2 Protocol-Specific RCT.”
• For AAL1, Section 30.10.2.2.3, “AAL1 Protocol-Specific RCT.”
• For AAL0, Section 30.10.2.2.4, “AAL0 Protocol-Specific RCT.”</td></tr>
<tr><td>Offset + 0x10</td></tr>
<tr><td>Offset + 0x12</td></tr>
<tr><td>Offset + 0x14</td></tr>
<tr><td>Offset + 0x16</td></tr>
<tr><td>Offset + 0x18</td></tr>
<tr><td>Offset + 0x1A</td><td colspan="16">MRBLR</td></tr>
<tr><td>Offset + 0x1C</td><td colspan="2">—</td><td colspan="6">PMT</td><td colspan="8">RBD_BASE</td></tr>
<tr><td>Offset + 0x1E</td><td colspan="12">RBD_BASE</td><td colspan="3">—</td><td>PM</td></tr>
</table>

Figure 30-26. Receive Connection Table (RCT) Entry

Table 30-15 describes RCT fields.

Table 30-15. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering—used for data buffers. 00 Reserved 01 Munged little endian 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffers bus 0 Data buffers reside on the 60x bus. 1 Reserved
	7	BIB	BD, interrupt queues, free buffer pool and external SRTS logic bus 0 Reside on the 60x bus 1 Reserved
	8	—	Reserved, should be cleared.
	9	BUFM	Buffer mode. (AAL5 only) See Section 30.10.5.3, “ATM Controller Buffers.” 0 Static buffer allocation mode. Each BD is associated with a dedicated buffer. 1 Global buffer allocation mode. Free buffers are fetched from global free buffer pools.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12–13	—	Reserved, should be cleared.
	14–15	INTQ	Points to one of four interrupt queues available
0x02	0	—	Internal use only. Should be cleared.
	1	INF	(AAL5 only) Indicates the receiver state. Initialize to 0 0 In idle state 1 In AAL5 frame reception state
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	(AAL5 only). Controls ABR flow. 0 ABR flow control is disabled. 1 ABR flow control is enabled.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol. Refer to Chapter 31, “AAL2 Protocol.” 101 AAL1_CES—Refer to Chapter 31, “ATM AAL1 Circuit Emulation Service.” All others reserved.

Table 30-15. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 MPC8272 time stamp timer is sampled and written to this field. See Section 13.3.7, “RISC Time-Stamp Control Register (RTSCR).”
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel’s current BD. User-initialized to 0; updated by the CP.
0x0E-0x18		—	Protocol-specific area
0x1A	—	MRBLR	Maximum receive buffer length. Used in both static and dynamic buffer allocation.
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	RBD_BASE	RxBD base. Points to the first BD in the channel’s RxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zeros.
0x1E	0–11		
	12–14	—	Reserved, should be cleared.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC 1 Perform performance monitoring for this VC. Whenever a cell is received for this VC the performance monitoring table that its code is written in the PMT field is updated.

30.10.2.2.1 AAL5 Protocol-Specific RCT

Figure 30-27 shows the AAL5 protocol-specific area of an RCT entry.

	0				15
Offset + 0x0E	TML				
Offset + 0x10	RX CRC				
Offset + 0x12					
Offset + 0x14	RBDCNT				
Offset + 0x16	—				
Offset + 0x18	—	RXBM	RXFM	—	BPOOL

Figure 30-27. AAL5 Protocol-Specific RCT

Table 30-16 describes AAL5 protocol specific RCT fields.

Table 30-16. RCT Settings (AAL5 Protocol-Specific)

Offset	Bits	Name	Description
0x0E	—	TML	Total message length. This field is used by the CP.
0x10	—	RxCRC	CRC32 temporary result

Table 30-16. RCT Settings (AAL5 Protocol-Specific) (continued)

Offset	Bits	Name	Description
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. RBDCNT is initialized with MRBLR whenever the CP opens a new buffer.
0x16	—	—	Reserved, should be cleared.
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask. Determines whether the receive buffer event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The event is enabled for this channel.
	9	RXFM	Receive frame interrupt mask. Determines whether the receive frame event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (RXF event is not sent to the interrupt queue.) 1 The event is enabled for this channel.
	10–13	—	Reserved, should be cleared.
	14–15	BPOOL	Buffer pool. Global buffer allocation mode only. Points to one of four free buffer pools. See Section 30.10.5.2.4, “Free Buffer Pool Parameter Tables.”

30.10.2.2.2 AAL5-ABR Protocol-Specific RCT

Figure 30-28 shows the AAL5-ABR protocol-specific area of an RCT entry.

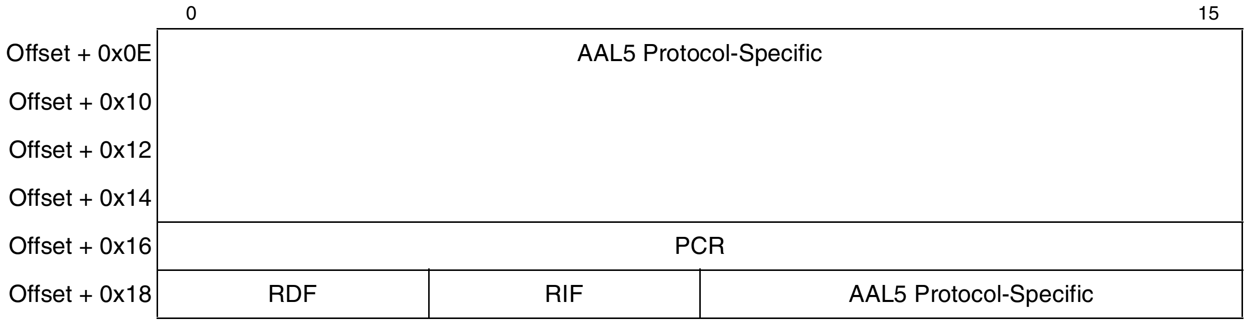


Figure 30-28. AAL5-ABR Protocol-Specific RCT

Table 30-17 describes AAL5-ABR protocol-specific RCT fields.

Table 30-17. ABR Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	—	—	AAL5 protocol specific
0x16	—	PCR	Peak cell rate. The peak number of cells per second of the current ABR channel. The ACR (allowed cell rate) never exceeds this value. PCR uses the ATMF TM 4.0 floating-point format.

Table 30-17. ABR Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	0–3	RDF	Rate decrease factor for the current ABR channel. Controls the decrease in cell transmission rate upon receipt of a backward RM cell. RDF represents a negative exponent of two, that is, the decrease factor = 2^{-RDF}. The decrease factor ranges from 1/32768 (RDF=0xF) to 1 (RDF=0).
	4–7	RIF	Rate increase factor of the current ABR channel. Controls the increase in the cell transmission rate upon receipt of a backward RM cell. RIF represents a negative exponent of two, that is, the increase factor = 2^{-RIF}. The increase factor ranges from 1/32768 (RIF=0xF) to 1 (RIF=0).
	8–15	—	AAL5 protocol specific

30.10.2.2.3 AAL1 Protocol-Specific RCT

Figure 30-29 shows the AAL1 protocol-specific area of an RCT entry.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x0E</td><td colspan="8">—</td><td>PFM</td><td>SRT</td><td>INVE</td><td>STF</td><td colspan="4">—</td></tr>
<tr><td>Offset + 0x10</td><td colspan="4">SRTS_TMP</td><td colspan="4">—</td><td colspan="4">—</td><td colspan="4">SRTS_DEV</td></tr>
<tr><td>Offset + 0x12</td><td colspan="2">—</td><td colspan="6">Valid Octet Size (VOS)</td><td>SPV</td><td colspan="7">Structured Pointer (SP)</td></tr>
<tr><td>Offset + 0x14</td><td colspan="16">RBDCNT</td></tr>
<tr><td>Offset + 0x16</td><td colspan="13">—</td><td colspan="3">SN</td></tr>
<tr><td>Offset + 0x18</td><td colspan="4">—</td><td>SNEM</td><td colspan="3">—</td><td>RXBM</td><td colspan="7">—</td></tr>
</table>

Figure 30-29. AAL1 Protocol-Specific RCT

Table 30-18 describes AAL1 protocol-specific RCT fields.

Table 30-18. AAL1 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–7	—	Reserved, should be cleared.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8272 supports clock recovery using an external SRTS PLL. The MPC8272 tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the CP writes a valid SRTS to external logic. 0 SRTS mode is not used 1 SRTS mode is used
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12–15	—	Reserved, should be cleared.
0x10	0–3	SRTS_TMP	Used by the CP to store the received SRTS code. After a cell with SN = 7 is received, the CP writes the SRTS code to the external SRTS device.
	4–11	—	Reserved, should be cleared.
	12–15	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.
0x12	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only.
	8	SPV	Structured pointer valid. Should be user-initialized user to zero. Structured format only.
	9–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. This field should be initialized by the user to zero. Used in structured format only.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. Initialized with MRBLR whenever the CP opens a new buffer.
0x16	0–12	—	Reserved, should be cleared.
	13–15	SN	Sequence number. Used by the CP to check incoming cell’s sequence number.

Table 30-18. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
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.

30.10.2.2.4 AAL0 Protocol-Specific RCT

Figure 30-30 shows the layout for the AAL0 protocol-specific RCT.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x0E</td><td colspan="8">—</td><td>0</td><td>1</td><td>INVE</td><td colspan="5">—</td></tr>
<tr><td>Offset + 0x10</td><td colspan="16" rowspan="4">—</td></tr>
<tr><td>Offset + 0x12</td></tr>
<tr><td>Offset + 0x14</td></tr>
<tr><td>Offset + 0x16</td></tr>
<tr><td>Offset + 0x18</td><td colspan="8">—</td><td>RXBM</td><td colspan="7">—</td></tr>
</table>

Figure 30-30. AAL0 Protocol-Specific RCT

Table 30-19 describes AAL0 protocol specific RCT fields.

Table 30-19. AAL0-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–7	—	Reserved, should be cleared.
	8–9	0b01	Must be programmed to 0b01 for AAL0
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11–15	—	Reserved, should be cleared.
0x10	—	—	Reserved, should be cleared.

Table 30-19. AAL0-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is masked. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

30.10.2.2.5 AAL2 Protocol-Specific RCT

Refer to Section 31.4.4.1, “AAL2 Protocol-Specific RCT.”

30.10.2.3 Transmit Connection Table (TCT)

Figure 30-31 shows the format of an TCT entry.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x00</td><td colspan="2">—</td><td>GBL</td><td colspan="2">BO</td><td>—</td><td>DTB</td><td>BIB</td><td>AVCF</td><td>—</td><td colspan="2">ATT</td><td>CPUU</td><td>VCON</td><td colspan="2">INTQ</td></tr>
<tr><td>Offset + 0x02</td><td>—</td><td>INF</td><td colspan="10">—</td><td>ABRF</td><td colspan="3">AAL</td></tr>
<tr><td>Offset + 0x04</td><td colspan="16" rowspan="2">Tx Data Buffer Pointer (TXDBPTR)</td></tr>
<tr><td>Offset + 0x06</td></tr>
<tr><td>Offset + 0x08</td><td colspan="16">TBDCNT</td></tr>
<tr><td>Offset + 0x0A</td><td colspan="16">TBD_OFFSET</td></tr>
<tr><td>Offset + 0x0C</td><td colspan="8">Rate Remainder</td><td colspan="8">PCR Fraction</td></tr>
<tr><td>Offset + 0x0E</td><td colspan="16">PCR</td></tr>
<tr><td>Offset + 0x10</td><td colspan="16" rowspan="3">Protocol Specific
• For AAL5,Section 30.10.2.3.1, “AAL5 Protocol-Specific TCT.”
• For AAL2, Section 31.3.5.1, “AAL2 Protocol-Specific TCT.”
• For AAL1, Section 30.10.2.3.2, “AAL1 Protocol-Specific TCT.”
• For AAL0, Section 30.10.2.3.3, “AAL0 Protocol-Specific TCT.”</td></tr>
<tr><td>Offset + 0x12</td></tr>
<tr><td>Offset + 0x14</td></tr>
<tr><td>Offset + 0x16</td><td colspan="16">APC Linked Channel (APCLC)</td></tr>
<tr><td>Offset + 0x18</td><td colspan="16" rowspan="2">ATM Cell Header (VPI,VCI,PTI,CLP)</td></tr>
<tr><td>Offset + 0x1a</td></tr>
<tr><td>Offset + 0x1C</td><td colspan="2">—</td><td colspan="6">PMT</td><td colspan="8">TBD_BASE</td></tr>
<tr><td>Offset + 0x1E</td><td colspan="12">TBD_BASE</td><td>BNM</td><td>STPT</td><td>IMK</td><td>PM</td></tr>
</table>

Figure 30-31. Transmit Connection Table (TCT) Entry

Table 30-20 describes general TCT fields.

Table 30-20. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering. This field is used for data buffers. 00 Reserved 01 PowerPC little endian 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffer bus 0 Reside on the 60x bus 1 Reserved
	7	BIB	BD, interrupt queue and external SRTS logic bus 0 Reside on the 60x bus 1 Reserved **Note:** When using AAL5, AAL1 CES in UDC mode, BDs and data should be placed on the same bus (TCT[DTB]=TCT[BIB]).
	8	AVCF	Auto VC off. Determines the behavior of the APC when the last buffer associated with this VC has been sent and no more ready buffers are in the VC’s TxBD table. 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears TCT[VCON]. **Note:** When over-subscribing UBR or UBR+ channels, set AVCF so that the CPM does not become overloaded polling non-active VCs.
	9	—	Reserved, should be cleared.
	10–11	ATT	ATM traffic type 00 Peak cell-rate pacing. The host must initialize PCR and the PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and minimum cell rate pacing (UBR+ traffic). The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved
	12	CPUU	CPCS-UU+CPI insertion (used for AAL5 only). 0 CPCS-UU+CPI insertion disabled. The transmitter clears the CPCS-UU+CPI fields. 1 CPCS-UU+CPI insertion enabled. The transmitter reads the CPCS-UU+CPI (16-bit entry) from external memory. It should be placed after the end of the last buffer (it should not be included in the buffer length).
	13	VCON	Virtual channel is on Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPT] (stop transmit), the CP deactivates this channel and clears VCON when the channel is next encountered in the APC scheduling table. The host can issue another ATM TRANSMIT command only after the CP clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 30-20. 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
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	Used for AAL5 Only. 0 ABR flow control is disabled. 1 ABR flow control is enabled.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol 100 AAL2—ATM adaptation layer 2 protocol. Refer to Chapter 31, "AAL2 Protocol." 101 AAL1_CES—Refer to Chapter 31, "ATM AAL1 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. Should be cleared initially.
0x0C	0–7	Rate reminder	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared initially.
	8–15	PCR fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC linked channel. Used by the CP. Initialize to 0 (null pointer).
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.

Table 30-20. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	TBD_BASE	TxBD base. Points to the first BD in the channel’s TxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zero.
0x1E	0–11		
	12	BNM	Buffer-not-ready interrupt mask. Can be changed on-the-fly. 0 The transmit buffer-not-ready event of this channel is masked. (TBNR event is not sent to the interrupt queue.) 1 The buffer-not-ready event of this channel is enabled.
	13	STPT	Stop transmit. Should be cleared initially. When the host sets this bit, the CP deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. Note that for AAL5 if STPT is set and frame transmission is already started (TCT[INF]=1), an abort indication will be sent (last cell with zero length field).
0x1E	14	IMK	Interrupt mask. Can be changed on-the-fly. 0 The transmit buffer event of this channel is masked. (TXB event is not sent to the interrupt queue.) 1 The transmit buffer event of this channel is enabled.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

30.10.2.3.1 AAL5 Protocol-Specific TCT

Figure 30-32 shows the AAL5 protocol-specific TCT.

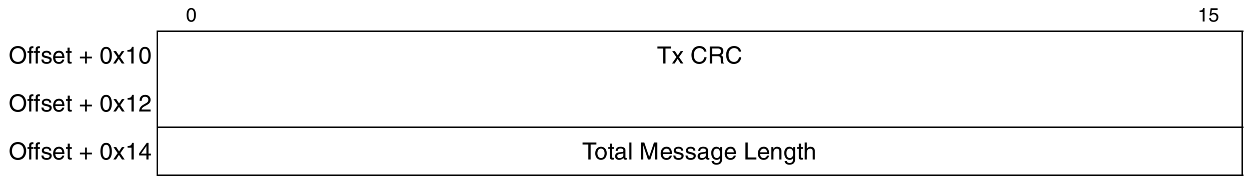


Figure 30-32. AAL5 Protocol-Specific TCT

Table 30-21 describes AAL5 protocol-specific TCT fields.

Table 30-21. AAL5-Specific TCT Field Descriptions

Offset	Name	Description
0x10	Tx CRC	CRC32 temporary result
0x14	Total message length	This field is used by the CP.

30.10.2.3.2 AAL1 Protocol-Specific TCT

Figure 30-33 shows the AAL1 protocol-specific TCT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x10	—		Valid Octet Size (VOS)						PFM	SRT	SPF	STF	—	SN		
Offset + 0x12	SRTS_DEV				Block Size											
Offset + 0x14	SRTS_TMP				Structured Pointer (SP)											

Figure 30-33. AAL1 Protocol-Specific TCT

Table 30-22 describes AAL1 protocol-specific TCT fields.

Table 30-22. AAL1 Protocol-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1–47 are valid; for structured service, values 1–46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8272 supports SRTS generation using external logic. If this mode is enabled, the MPC8272 reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The MPC8272 reads the new SRTS from external logic every eight cells. 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPF	Structured pointer flag. Indicates that a structured pointer has been inserted in the current block. The user should initialize this field to zero. Used by the CP only.
	11	STF	Structured format 0 Unstructured format is used 1 Structured format is used
	12	—	Reserved, should be cleared.
	13–15	SN	Sequence number field. Used by the CP to check the incoming cells SN. Should be cleared initially.
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28–31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum).
0x14	0–3	SRTS_TMP	Before a cell with SN = 1 is sent, the CP reads the SRTS code from external SRTS logic, writes it to SRTS_TMP and inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. Should be cleared initially. Structured format only.

30.10.2.3.3 AAL0 Protocol-Specific TCT

Figure 30-34 shows the AAL0 protocol-specific TCT.

	0 7	8	9	10	11	12 15
Offset + 0x10	—	0	CR10	—	ACHC	—
Offset + 0x12	—					
Offset + 0x14						

Figure 30-34. AAL0 Protocol-Specific TCT

Table 30-23 describes AAL0 protocol-specific TCT fields.

Table 30-23. AAL0-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–7	—	Reserved, should be cleared.
	8	0	Must be 0
	9	CR10	CRC-10 0 CRC10 insertion is disabled. 1 CRC10 insertion is enabled.
	10	—	Reserved, should be cleared.
	11	ACHC	ATM cell header change 0 Normal operation ATM cell header is taken from AAL0 buffer 1 VPI/VCI (28 bits) are taken from TCT
	12–15	—	Reserved, should be cleared.
0x12–0x14	—	—	Reserved, should be cleared.

30.10.2.3.4 AAL2 Protocol-Specific TCT

Refer to Section 31.3.5.1, “AAL2 Protocol-Specific TCT.”

30.10.2.3.5 VBR Protocol-Specific TCTE

Figure 30-35 shows the VBR protocol-specific TCTE.

	0	1 … 7	8 … 15
Offset + 0x00	SCR		
Offset + 0x02	Burst Tolerance (BT)		
Offset + 0x04	Out of Buffer Rate (OOBR)		
Offset + 0x06	Sustain Rate Remainder (SRR)		SCR Fraction (SCRF)
Offset + 0x08	Sustain Rate (SR)		
Offset + 0x0A			
Offset + 0x0C	VBR2	—	
Offset + 0x0E-1E	—		

Figure 30-35. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 30-24 describes VBR protocol-specific TCTE fields.

Table 30-24. VBR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	SCR	Sustain cell rate. Holds the sustain cell rate (in slots) permitted for this channel according to the traffic contract. To pace the channel's sustain cell rate, the APC performs a continuous-state leaky bucket algorithm (GCRA).
0x02	—	BT	Burst tolerance. Holds the burst tolerance permitted for this channel according to the traffic contract. The relationship between the BT and the maximum burst size (MBS) is BT=(MBS-2) × (SCR-PCR) + SCR.
0x04	—	OOBR	Out-of-buffer rate. In out of buffer state (when the transmitter tries to open TxBD whose R bit is not set) the APC reschedules the current channel according to OOBR rate.
0x06	0–7	SRR	Sustain rate remainder. Holds the sustain rate remainder after adding the pace fraction field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state. Initialized to 0.
	8–15	SCRF	Holds the sustain cell rate fraction of this channel in units of 1/256 slot.
0x08	—	SR	Sustain rate. Used by the APC to hold the sustain rate after adding the pace field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state.
0x0C	0	VBR2	VBR type 0 Regular VBR. CLP=0+1 cells are rescheduled by PCR or SCR according to the GCRA state. 1 VBR type 2. CLP=0 cells are rescheduled by PCR or SCR according to the GCRA state. CLP=1 cells are rescheduled by PCR.
	1–15	—	Reserved, should be cleared.
0x0E– 0x1E	—	—	Reserved, should be cleared.

30.10.2.3.6 UBR+ Protocol-Specific TCTE

Figure 30-36 shows the UBR+ protocol-specific TCTE.

	0 7	8 15
Offset + 0x00	MCR	
Offset + 0x02	—	MCR Fraction (MCRF)
Offset + 0x04	Maximum Delay Allowed (MDA)	
Offset + 0x06–0x1E	—	

Figure 30-36. UBR+ Protocol-Specific TCTE

Table 30-25 describes UBR+ protocol-specific TCTE fields.

Table 30-25. UBR+ Protocol-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	MCR	Minimum cell rate for this channel. MCR is in units of APC time slots.
0x02	0–7	—	Reserved, should be cleared.
	8–15	MCRF	Minimum cell rate fraction. Holds the minimum cell rate fraction of this channel in units of 1/256 slot.
0x04	—	MDA	Maximum delay allowed. The maximum time-slot service delay allowed for this priority level before the APC reduces the scheduling rate from PCR to MCR.
0x06–0x1E	—	—	Reserved, should be cleared.

30.10.2.3.7 ABR Protocol-Specific TCTE

Figure 30-37 shows the ABR protocol-specific TCTE.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x00</td><td colspan="16">ER-TA</td></tr>
<tr><td>Offset + 0x02</td><td colspan="16">CCR-TA</td></tr>
<tr><td>Offset + 0x04</td><td colspan="16">MCR-TA</td></tr>
<tr><td>Offset + 0x06</td><td>TUAR</td><td>—</td><td>CI-TA</td><td>NI-TA</td><td colspan="3">—</td><td>CP-TA</td><td colspan="2">—</td><td>CI-VC</td><td colspan="5">—</td></tr>
<tr><td>Offset + 0x08</td><td colspan="16">MCR</td></tr>
<tr><td>Offset + 0x0A</td><td colspan="16">UNACK</td></tr>
<tr><td>Offset + 0x0C</td><td colspan="16">ACR</td></tr>
<tr><td>Offset + 0x0E</td><td>ACRC</td><td colspan="15">—</td></tr>
<tr><td>Offset + 0x10</td><td colspan="16" rowspan="2">RM Cell Time Stamp (RCTS)</td></tr>
<tr><td>Offset + 0x12</td></tr>
<tr><td>Offset + 0x14</td><td>FRST</td><td colspan="3">—</td><td colspan="4">CDF</td><td colspan="8">COUNT</td></tr>
<tr><td>Offset + 0x16</td><td colspan="16">ICR</td></tr>
<tr><td>Offset + 0x18</td><td colspan="16">CRM</td></tr>
<tr><td>Offset + 0x1A</td><td colspan="16">ADTF</td></tr>
<tr><td>Offset + 0x1C</td><td colspan="16">ER</td></tr>
<tr><td>Offset + 0x1E</td><td colspan="16">ER-BRM</td></tr>
</table>

Figure 30-37. ABR Protocol-Specific TCTE

Table 30-26 describes ABR-specific TCTE fields.

Table 30-26. ABR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	ER-TA	Explicit rate–turn-around cell. Holds the ER of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's ER.
0x02	—	CCR-TA	Current cell rate–turn-around cell. Holds the CCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's CCR.
0x04	—	MCR-TA	Minimum cell rate–turn-around cell. Holds the MCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell is turned around, this field is overwritten by the new RM cell's MCR.
0x06	0	TUAR	Turn-around flag. The CP sets TUAR to indicate that a new F-RM cell was received, which causes the transmitter to send a B-RM cell whenever the ABR flow control permits. Should be cleared initially.
	1	—	Reserved, should be cleared.
	2	CI-TA	Congestion indication–turn-around cell. Holds the CI of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, CI-TA is overwritten by the new RM cell's CI.

Table 30-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
	3	NI-TA	No increase–turn-around cell. Holds the NI of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, NI-TA is overwritten by the new RM cell's NI.
	4–6	—	Reserved, should be cleared.
	7	CP-TA	Cell loss priority–turn-around cell. Holds the CLP of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, CP-TA is overwritten by the new RM cell's CLP.
	8–9	—	Reserved, should be cleared.
	10	CI-VC	Congestion indication -VC. Holds the EFCI (explicit forward congestion indication) of the last user data cell. The CI bit of the turned around RM cell is ORed with the CI-VC. Should be cleared initially.
	11–15	—	Reserved, should be cleared.
0x08	—	MCR	Minimum cell rate Holds the minimum number of cells/sec of the current ABR channel. Uses the ATMF TM 4.0 floating-point format.
0x0A	—	UNACK	Used by the CP to count F-RM cells sent in an absence of received B-RM cells. Should be cleared initially.
0x0C	—	ACR	Allowed cell rate The cells per second allowed for the current ABR channel. Uses the ATMF TM 4.0 floating-point format. Initialize with ICR.
0x0E	0	ACRC	ACR change. Indicates a change in ACR. Initialize to one.
	1–15	—	Reserved, should be cleared.
0x10	—	RCTS	RM cell time stamp. Used exclusively by the CP. Initialize to zero.
0x14	0	FRST	First turn. Used exclusively by the CP. Indicates the first turn of a backward RM cell, which has priority over a data cell. Initialized to 0.
	1–3	—	Reserved, should be cleared.
	4–7	CDF	Cutoff decrease factor. Controls the decrease in the ACR associated with missing B-RM cells feedback. CDF represents a negative exponent of two, that is, the cutoff decrease factor = 2^{-CDF}. The cutoff decrease factor ranges from 1/64 (CDF=0b0110) to 1 (CDF=0b0000). All other CDF values falling outside this range are invalid.
	8–15	COUNT	Count. Used only by the CP. Holds the number of cells sent since the last forward RM cell. Initialize with Nrm (in the parameter RAM).
0x16	—	ICR	Initial cell rate. The number of cells per second of the current ABR channel. The channel's ACR is initialized with ICR. ICR uses the ATMF TM 4.0 floating-point format.
0x18	—	CRM	Missing RM cells count. Limits the number of forward RM cells that may be sent in the absence of received backward RM cell. The CRM is in units of cells.
0x1A	—	ADTF	ADTF–ACR decrease time factor. The ADTF period is 500 ms as defined in the TM 4.0. The ADTF value is defined by the system clock and the time stamp timer prescaler; see Section 13.3.7, "RISC Time-Stamp Control Register (RTSCR)." For a time stamp prescaler of 1 µs, ADTF should be programmed to 500m/(1µs × 1024)= 488.

Table 30-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	—	ER	Explicit rate. Holds the explicit rate value (in cells/sec) of the current ABR channel. ER is copied to the F-RM cell ER field. The user usually initializes this field to PCR. ER uses the ATMF TM 4.0 floating-point format.
0x1E	—	ER-BRM	Explicit rate-backward RM cell. Holds the maximum explicit rate value (in cells/sec) allowed for B-RM cells. The ER-TA field that is inserted to each B-RM cell is limited by this value. ER-BRM uses the ATMF TM 4.0 floating-point format.

30.10.3 OAM Performance Monitoring Tables

The OAM performance monitoring tables include performance monitoring block test parameters, as shown in Figure 30-38. Each block test needs a 32-byte performance monitoring table in the dual-port RAM. In the connection’s RCT and TCT, the user allocates an OAM performance table to a VCC or VPC. See Section 30.6.6, “Performance Monitoring.” PMT_BASE in the parameter RAM points to the base address of the tables. The starting address of each PM table is given by PMT_BASE + RCT/TCT[PMT] × 32.

	0	1	2 4	5 7	8 15
Offset + 0x00	**FMCE**	**TSTE**	—	**BLCKSIZE**	
Offset + 0x02	—			TX Cell Count (**TCC**)	
Offset + 0x04	**TUC1**				
Offset + 0x06	**TUC0**				
Offset + 0x08	**BEDC0+1-Tx**				
Offset + 0x0A	**BEDC0+1-RX**				
Offset + 0x0C	**TRCC1**				
Offset + 0x0E	**TRCC0**				
Offset + 0x10	—				**SN-FMC**
Offset + 0x12	—				
Offset + 0x14	PM CELL HEADER (VPI,VCI,PTI,CLP)				
Offset + 0x16					
Offset + 0x18	—				
Offset + 0x1A					
Offset + 0x1C					
Offset + 0x1E					

Figure 30-38. OAM Performance Monitoring Table

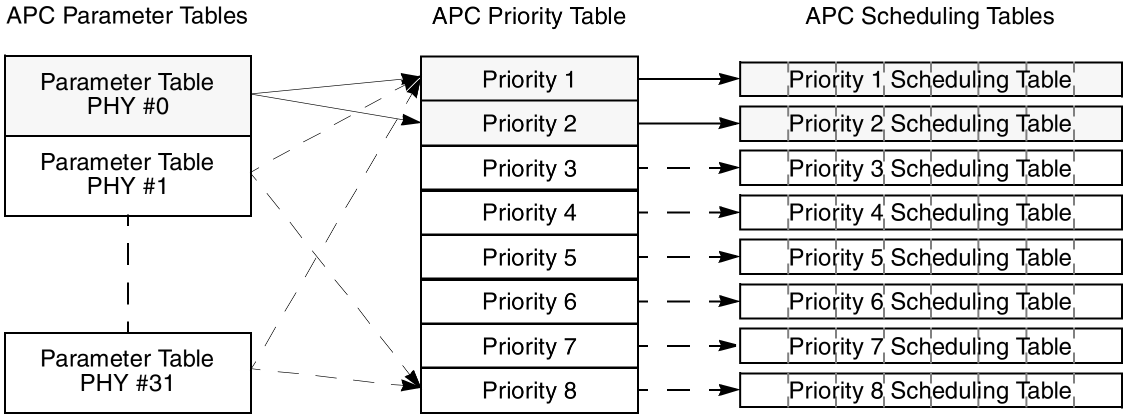
Table 30-27 describes fields in the performance monitoring table.

Table 30-27. 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 1s. 1 The value of the time stamp timer is inserted into the time stamp field of the FMC.
	2–4	—	Reserved, should be cleared.
	5–15	**BLCKSIZE**	Performance monitoring block size ranging from 1 to 2,047 cells.
0x02	0–4	—	Reserved, should be cleared.
	5–15	**TCC**	TX cell count. Used by the CP to count data cells sent. Initialize to zero.
0x04	—	**TUC1**	Total user cell 1. Count of CLP = 1 user cells (modulo 65,536) sent. Should be cleared initially.
0x06	—	**TUC0**	Total user cell 0. Count of CLP = 0 user cells (modulo 65,536) sent. Should be cleared initially.
0x08	—	**BEDC0+1-Tx**	Block error detection code 0+1–transmitted cells. Even parity over the payload of the block of user cells sent since the last FMC. Should be cleared initially.
0x0A	—	**BEDC0+1-RX**	Block error detection code 0+1–received cells. Even parity over the payload of the block of user cells received since the last FMC. Should be cleared initially.
0x0C	—	**TRCC1**	Total received cell 1. Count of CLP = 1 user cells (modulo 65,536) received. Should be cleared initially.
0x0E	—	**TRCC0**	Total received cell 0. Count of CLP = 0 user cells (modulo 65,536) received. Should be cleared initially.
0x10	0–7	—	Reserved, should be cleared.
	8–15	**SN-FMC**	Sequence number of the last FMC sent. Should be cleared initially.
0x12	—	—	Reserved, should be cleared.
0x14	—	**PMCH**	PM cell header. Holds the ATM cell header of the FMC, BRC to be inserted by the CP into the Tx cell flow.
0x18– 0x1E	—	—	Reserved, should be cleared.

30.10.4 APC Data Structure

The APC data structure consists of three elements: the APC parameter tables for the PHY devices, the APC priority table, and the APC scheduling tables. See Figure 30-39.


APC Parameter Tables
APC Priority Table
APC Scheduling Tables
Parameter Table PHY #0
Parameter Table PHY #1
Parameter Table PHY #31
Priority 1
Priority 2
Priority 3
Priority 4
Priority 5
Priority 6
Priority 7
Priority 8
Priority 1 Scheduling Table
Priority 2 Scheduling Table
Priority 3 Scheduling Table
Priority 4 Scheduling Table
Priority 5 Scheduling Table
Priority 6 Scheduling Table
Priority 7 Scheduling Table
Priority 8 Scheduling Table

Note: The shaded areas represent the active structures for an example implementation of PHY #0 with two priorities. (The unshaded areas and dashed arrows represent unused structures.)

Figure 30-39. ATM Pace Control Data Structure

30.10.4.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 30-28, holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The table resides in the dual-port RAM. The parameter APCP_BASE, described in Section 30.10.1, "Parameter RAM," points to the base address of PHY#0's parameter table.

For multiple PHYs, the table structure is duplicated. Each table resides in 32 bytes of memory. The starting address of each APC parameter table is given by APCP_BASE + PHY# × 32. Note however that in slave mode with multiple PHYs, the parameter table always resides at APCP_BASE regardless of the PHY address.

Table 30-28. APC Parameter Table

Offset[1]	Name	Width	Description
0x00	**APCL_FIRST**	Hword	Address of first entry in the priority table. Must be 8-byte aligned. User-initialized.
0x02	**APCL_LAST**	Hword	Address of last entry in the priority table. Must be 8-byte aligned. User-initialized as APCL_FIRST + 8 x (number_of_priorities – 1).
0x04	**APCL_PTR**	Hword	Address of current priority entry used by the CP. User-initialized with APCL_FIRST.
0x06	**CPS**	Byte	Cells per slot. Determines the number of cells sent per APC slot. See Section 30.3.2, "APC Unit Scheduling Mechanism." User-defined. (0x01 = 1 cell; 0xFF = 255 cells.) **Note:** If ABR is used, CPS must be a power of two.
0x07	**CPS_CNT**	Byte	Cells sent per APC slot counter. User-initialized to CPS; used by the CP.
0x08	**MAX_ITERATION**	Byte	Max iteration allowed. Number of scan iterations allowed in the APC. User-defined. This parameter limits the time spent in a single APC routine, thereby avoiding excessive APC latency.

Table 30-28. APC Parameter Table (continued)

Offset[1]	Name	Width	Description
0x09	**CPS_ABR**	Byte	ABR only. Cells per slot represented as a power of two. User-defined. (For example, if CPS is 1, CPS_ABR = 0x00; if CPS is 8, CPS_ABR = 0x03.)
0x0A	**LINE_RATE_ABR**	Hword	ABR only. The PHY line rate in cells/sec, represented in TM 4.0 floating-point format. User-defined.
0xC	**REAL_TSTP**	Word	Real-time stamp pointer used internally by the APC. Should be cleared initially.
0x10	**APC_STATE**	Word	Used internally by the APC. Should be cleared initially.

[1] Offset values are to APCP_BASE+PHY# × 32. However, in slave mode, the offset is from APCP_BASE regardless of the PHY address.

30.10.4.2 APC Priority Table

Each PHY’s APC priority table holds pointers to the APC scheduling table of each priority level. It resides in the dual-port RAM. The priority table can hold up to eight priority levels. Table 30-29 shows the structure of a priority table entry.

Table 30-29. APC Priority Table Entry

Offset	Name	Width	Description
0x00	APC_LEVi_BASE	Hword	APC level i base address. Pointer to the first slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x02	APC_LEVi_END	Hword	APC level i end address. Pointer to the last slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x04	APC_LEVi_RPTR	Hword	APC level i real-time/service pointers. APC table pointers used internally by the APC. Initialize both pointers to APC_LEVi_BASE.
0x06	APC_LEVi_SPTR	Hword	

30.10.4.3 APC Scheduling Tables

The APC uses APC scheduling tables (one table for each priority level) to schedule channel transmission. A scheduling table is divided into time slots, as shown in Figure 30-40. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

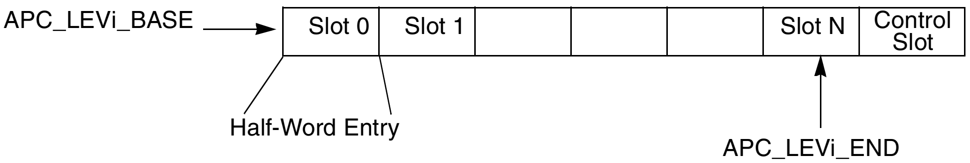


Figure 30-40. The APC Scheduling Table Structure

Slot N+1 is used as a control slot, as shown in Figure 30-41.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	**TCTE**	000_0000_0000_0000														

Figure 30-41. Control Slot

Table 30-30 describes control slot fields.

Table 30-30. 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. (No external VBR, ABR, UBR+ channels) 1 Channels in this scheduling table use external TCTE. (External VBR, ABR, UBR+ channels)
1–15	—	Reserved, should be cleared.

30.10.5 ATM Controller Buffer Descriptors (BDs)

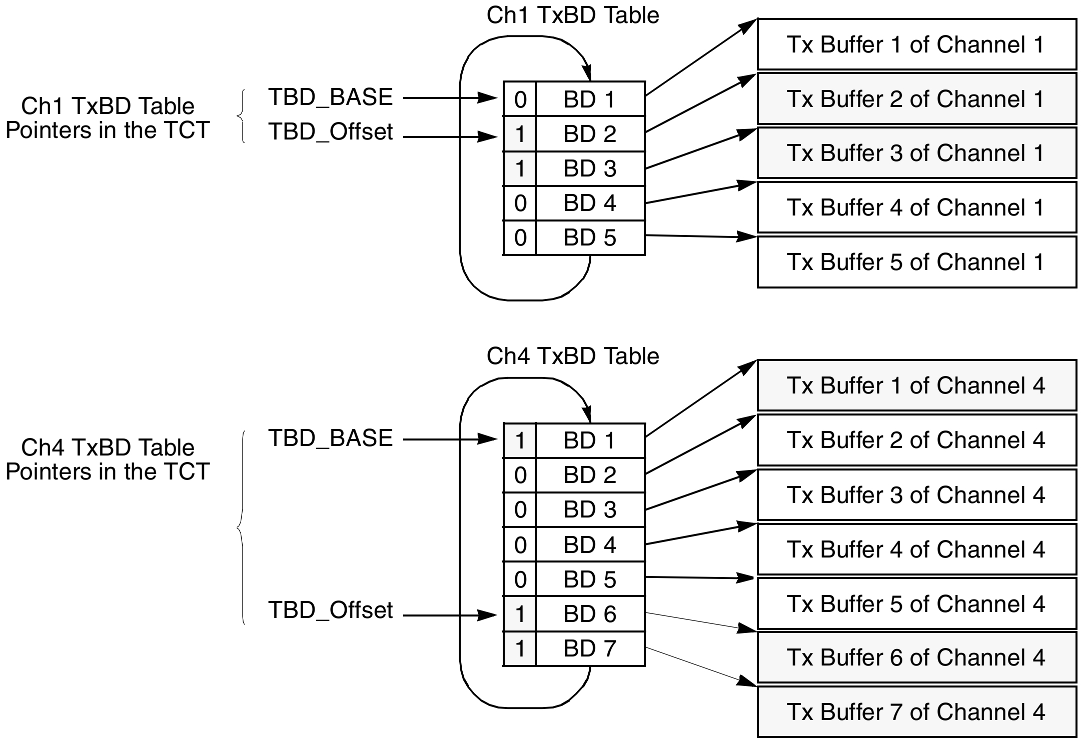
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- Static buffer allocation. In this mode, the user allocates dedicated buffers to each ATM channel (that is, the user associates each BD with one buffer). Static buffer allocation is useful when the connection rate is known and constant and when data must be reassembled in a particular memory space.
- Global buffer allocation. Available for AAL5 only. In this mode, buffer allocation is dynamic. The user allocates receive buffers and places them in global buffer pools. When the CP needs a receive buffer, it first fetches a buffer pointer from one of the global buffer pools and writes the pointer to the current RxBD. Global buffer allocation is optimized for allocating memory among many ATM channels with variable data rates, such as ABR channels.

30.10.5.1 Transmit Buffer Operation

The user prepares a table of BDs pointing to the buffers to be sent. The address of the first BD is put in the channel’s TCT[TBD_BASE]. The transmit process starts when the core issues an ATM TRANSMIT command. The CP reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the CP increments TBD_Offset, which holds the offset from TBD_BASE to the current BD. It then reads the next BD in the table. If the BD is ready (TxBD[R] = 1), the CP continues sending. If the current BD is not ready, the CP polls the ready bit at the channel rate unless TCT[AVCF] = 1, in which case the CP removes the channel from the APC and clears TCT[VCON]. The core must issue a new ATM TRANSMIT command to restart transmission.

Figure 30-42 shows the ready bit in the TxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are ready to be sent; unshaded buffers are waiting to be prepared.

Figure 30-42. Transmit Buffers and BD Table Example

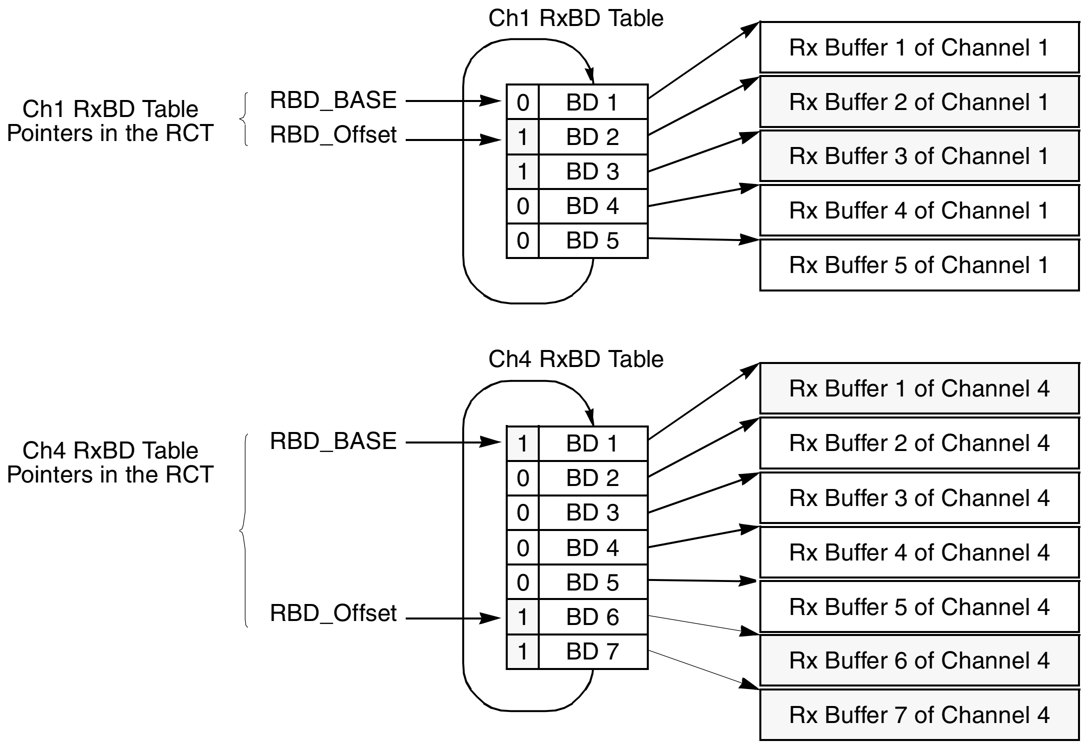
30.10.5.2 Receive Buffer Operation

For AAL5 channels, the user should choose to operate in static buffer allocation or in global buffer allocation by writing to RCT[BUFM]. AAL0 channels must use static buffer allocation.

30.10.5.2.1 Static Buffer Allocation

The user prepares a table of BDs pointing to the receive buffers. The address of the first BD is put in the channel’s RCT[RBD_BASE]. When an ATM cell arrives, the CP opens the first BD in the table and starts filling its associated buffer with received data. When the current buffer is full, the CP increments RBD_Offset, which is the offset to the current BD from RBD_BASE, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP continues receiving. If the BD is not empty, a busy condition has occurred and a busy interrupt is sent to the event queue.

Figure 30-43 shows the empty bit in the RxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are empty; unshaded buffers are waiting to be processed.

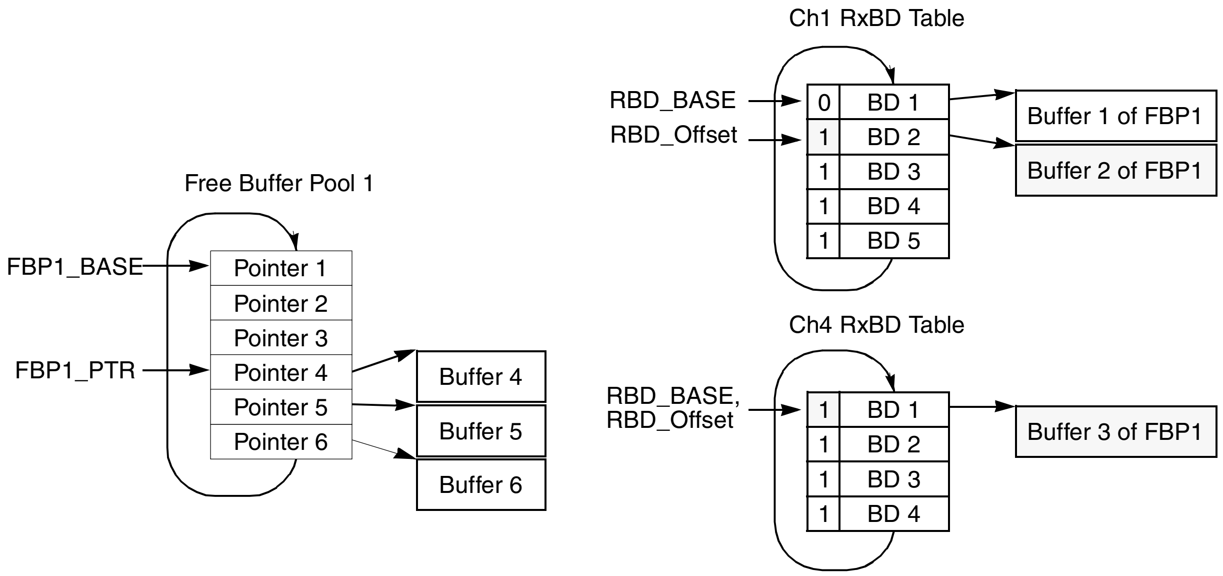
Figure 30-43. Receive Static Buffer Allocation Example

30.10.5.2.2 Global Buffer Allocation

The user prepares a table of BDs without assigning buffers to them (no buffer pointers). The address of the first BD is put into the channel’s RCT[RBD_BASE]. The user also prepares sets of free buffers (of size RCT[MRBLR]) in up to four free buffer pools (chosen in RCT[BPOOL]); see Section 30.10.5.2.3, “Free Buffer Pools.”

When an ATM cell arrives, the CP opens the first BD in the table, fetches a buffer pointer from the free buffer pool associated with this channel, and writes the pointer to RxBD[RXDBPTR], the receive data buffer pointer field in the BD. When the current buffer is full, the CP increments RBD_Offset, which is the offset from the RBD_BASE to the current BD, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP fetches another buffer pointer from the free buffer pool and reception continues. If the BD is not empty, a busy condition occurs and a busy interrupt is sent to the event queue specifying the ATM channel code. As software then processes each full buffer (RxBD[E] = 0), it sets RxBD[E] and copies the buffer pointer back to the free buffer pool.

Figure 30-44 shows two ATM channels’ BD tables and one free buffer pool. Both channels are associated with free buffer pool 1. The CP allocates the first two buffers of buffer pool 1 to channel 1 and the third to channel 4.

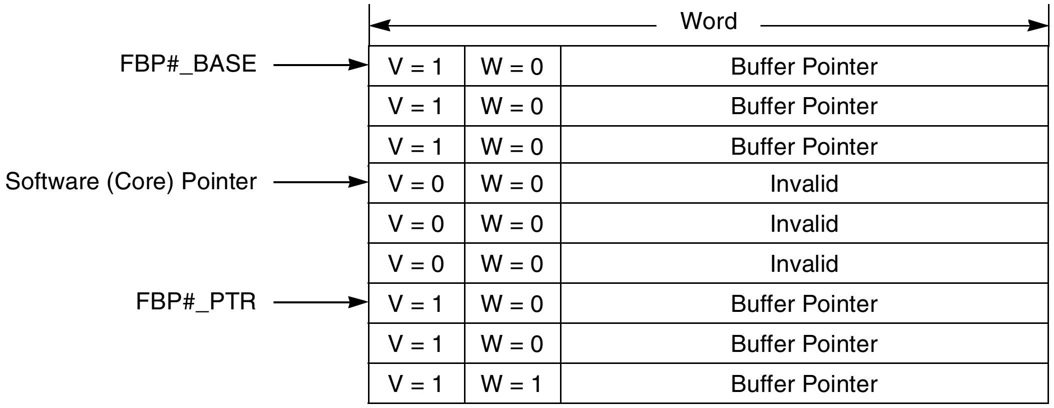


Notes: Buffers 2 and 3 are receiving data. After buffer 1 is processed, it can be returned to the pool.

Figure 30-44. Receive Global Buffer Allocation Example

30.10.5.2.3 Free Buffer Pools

As Figure 30-45 shows, when a buffer pointer is fetched from a pool, the CP clears the entry’s valid bit and increments FBP#_PTR. After the CP uses an entry with the wrap bit set (W = 1), it returns to the first entry in the pool. After a buffer pointer is returned to the pool, the user should set V to indicate that the entry is valid. If the CP tries to read an invalid entry (V = 0), the buffer pool is out of free buffers; the global-buffer-pool-busy event is then set in FCCE[GBPB] and a busy interrupt is sent to the interrupt queue specifying the ATM channel code associated with the pool.



V = 1	W = 0	Buffer Pointer
V = 1	W = 0	Buffer Pointer
V = 1	W = 0	Buffer Pointer
V = 0	W = 0	Invalid
V = 0	W = 0	Invalid
V = 0	W = 0	Invalid
V = 1	W = 0	Buffer Pointer
V = 1	W = 0	Buffer Pointer
V = 1	W = 1	Buffer Pointer

Figure 30-45. Free Buffer Pool Structure

Figure 30-46 describes the structure of a free buffer pool entry.

	0	1	2	3	4 — 15
Offset + 0x00	**V**	—	**W**	**I**	**Buffer Pointer (BP)**
Offset + 0x02	**Buffer Pointer (BP)**				

Figure 30-46. Free Buffer Pool Entry

Table 30-31 describes free buffer pool entry fields.

Table 30-31. Free Buffer Pool Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid buffer entry. 0 This free buffer pool entry contains an invalid buffer pointer. 1 This free buffer pool entry contains a valid buffer pointer.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap bit. When set, this bit indicates the last entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3	**I**	Red-line interrupt. Can be used to indicate that the free buffer pool has reached a red line and additional buffers should be added to this pool to avoid a busy condition. 0 No interrupt is generated. 1 A red-line interrupt is generated when this buffer is fetched from the free buffer pool.
	4–15	**BP**	Buffer pointer. Points to the start address of the receive buffer. The four msbs are control bits, and the four msbs of the real buffer pointer are taken from the four msbs of the parameter FBP_ENTRY_EXT in the free buffer pool parameter table.
0x02	0–15		

30.10.5.2.4 Free Buffer Pool Parameter Tables

The free buffer pool parameters are held in parameter tables in the dual-port RAM; see Table 30-32. FBT_BASE in the parameter RAM points to the base address of these tables. Each of the four free buffer pools has its own parameter table with a starting address given by FBT_BASE + RCT[BPOOL] × 16.

Table 30-32. Free Buffer Pool Parameter Table

Offset[1]	Bits	Name	Description
0x00	—	**FBP_BASE**	Free buffer pool base. Holds the pointer to the first entry in the free buffer pool. FBP_BASE should be word aligned. User-defined.
0x04	—	**FBP_PTR**	Free buffer pool pointer. Pointer to the current entry in the free buffer pool. Initialize to FBP_BASE.
0x08	—	**FBP_ENTRY_EXT**	Free buffer pool entry extension. FBP_ENTRY_EXT[0–3] holds the four left bits of FBP_ENTRY. FBP_ENTRY_EXT[4–15] should be cleared. User-defined.

Table 30-32. Free Buffer Pool Parameter Table (continued)

Offset [1]	Bits	Name	Description
0x0A	0	**BUSY**	The CP sets this bit when it tries to fetch buffer pointer with V bit clear. FCCE[GBPB] is also set. Initialize to zero.
	1	**RLI**	Red-line interrupt. Set by the CP when it fetches a buffer pointer with I = 1. FCCE[GRLI] is also set. Initialize to zero.
	2–7	—	Reserved, should be cleared.
	8	**EPD**	Early packet discard. 0 Normal operation 1 AAL5 frames in progress are received, but new AAL5 frames associated with this pool are discarded. Can be used to implement EPD under core control.
	9–15	**—**	Reserved, should be cleared.
0x0C	—	**FBP_ENTRY**	Free buffer pool entry. Initialize with the first entry of the free buffer pool. Note that FBP_ENTRY must be reinitialized with the entry pointed to by FBP_PTR when a busy state occurs to reenable free buffer pool processing.

[1] Offset from FBT_BASE + RCT[BPOOL] × 16.

30.10.5.3 ATM Controller Buffers

Table 30-33 describes properties of the ATM receive and transmit buffers.

Table 30-33. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	**Alignment**	**Size**	**Alignment**
AAL5	Multiple of 48 octets (except last buffer in frame)	Burst-aligned	Any	No requirement
AAL1	At least 47 octets	Burst-aligned	At least 47 octets	No requirement
AAL0	52–64 octets.	Burst-aligned	52–64 octets.	No requirement

30.10.5.4 AAL5 RxBD

Figure 30-47 shows the AAL5 RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**E**	—	**W**	**I**	L	F	**CM**	—			CLP	CNG	ABRT	CPUU	LNE	CRE
Offset + 0x02	Data Length (DL)															
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**															
Offset + 0x06																

Figure 30-47. AAL5 RxBD

Table 30-34 describes AAL5 RxBD fields.

Table 30-34. 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 CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set in the event register when INT_CNT reaches the global interrupt threshold.
	4	L	Last in frame. Set by the ATM controller for the last buffer in a frame. 0 Buffer is not last in a frame 1 Buffer is last in a frame. ATM controller writes frame length in DL and updates the error flags.
	5	F	First in frame. Set by the ATM controller for the first buffer in a frame. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear the empty bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CLP	Cell loss priority. At least one cell associated with the current message was received with CLP = 1. May be set at the last buffer of the message.
	11	CNG	Congestion indication. The last cell associated with the current message was received with PTI middle bit set. CNG may be set at the last buffer of the message.
	12	ABRT	Abort message indication. The current message was received with Length field zero.
	13	CPUU	CPCS-UU+CPI indication. Set when the CPCS-UU+CPI field is non zero. CPUU may be set at the last buffer of the message.
	14	LNE	Rx length error. AAL5 CPCS-PDU length violation. May be set only for the last BD of the frame if the pad length is greater than 47 or less than zero octets.
	15	CRE	Rx CRC error. Indicates CRC32 error in the current AAL5 PDU. Set only for the last BD of the frame.

Table 30-34. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	—	DL	Data length. The number of octets written by the CP into this BD's buffer. It is written by the CP once the BD is closed. In the last BD of a frame, DL contains the total frame length.
0x04		**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in internal or external memory. It is recommended that the pointer be burst-aligned.

30.10.5.5 AAL1 RxBD

Figure 30-48 shows the AAL1 RxBD.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>Offset + 0x00</td><td><b>E</b></td><td>—</td><td><b>W</b></td><td><b>I</b></td><td>SNE</td><td>—</td><td><b>CM</b></td><td colspan="9">—</td></tr>
<tr><td>Offset + 0x02</td><td colspan="16">Data Length</td></tr>
<tr><td>Offset + 0x04</td><td colspan="16" rowspan="2"><b>Rx Data Buffer Pointer</b></td></tr>
<tr><td>Offset + 0x06</td></tr>
</table>

Figure 30-48. AAL1 RxBD

Table 30-35 describes AAL1 RxBD fields.

Table 30-35. 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 CP cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer is used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table overall space is constrained to 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4	SNE	Sequence number error. SNE is set when a sequence number error is detected in the current AAL1 CES buffer.
	5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation 1 The empty bit (RxBD[E]) is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–15	—	Reserved, should be cleared.
0x02	—	DL	Data length. The number of octets the CP writes into the buffer once its BD is closed.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. It is recommended that the pointer be burst-aligned.

30.10.5.6 AAL0 RxBD

Figure 30-49 shows the AAL0 RxBD.

	0	1	2	3	4 5	6	7 9	10	11	12 15
Offset + 0x00	**E**	—	**W**	**I**	—	**CM**	—	CRE	OAM	—
Offset + 0x02	Data Length (DL)/Channel Code (CC)									
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**									
Offset + 0x06										

Figure 30-49. AAL0 RxBD

Table 30-36 describes AAL0 RxBD fields.

Table 30-36. 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 CP does not use this BD again while E remains zero. 1 The Rx buffer is empty or reception is in progress. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of the current channel. After this buffer has been used, the CP will receive incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear the E bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CRE	Rx CRC error. Indicates a CRC10 error in the current AAL0 buffer. The CRE bit is considered an error only if the received cell had a CRC10 field in the cell payload.
	11	OAM	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an OAM cell. This cell is associated with the channel indicated by the channel code field (CC field).
	12–15	—	Reserved, should be cleared.
0x02	—	DL/CC	Data length/channel code. If RxBD[OAM] is set, this field functions as CC; otherwise, it is DL. Data length is the size in octets of this buffer (MRBLR value). Channel code specifies the channel code associated with this OAM cell.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. It is recommended that the pointer be burst-aligned.

30.10.5.7 AAL2 RxBD

Refer to Section 31.4.4.4, “CPS Receive Buffer Descriptor (RxBD).”

30.10.5.8 AAL5 User-Defined Cell—RxBD Extension

In user-defined cell mode, the AAL5 and AAL1 CES RxBDs are extended to 32 bytes; see Figure 30-50.

NOTE

> For AAL0, a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

Offset + 0x08	Extra Cell Header Used to store the user-defined cell’s extra cell header. The extra cell header can be 1–12 bytes long.
Offset + 0x14	Reserved (12 Bytes)

Figure 30-50. User-Defined Cell—RxBD Extension

30.10.5.9 AAL5 TxBDs

Figure 30-51 shows the AAL5 TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**R**	—	**W**	**I**	**L**	—	**CM**	—			**CLP**	**CNG**	—			
Offset + 0x02	**Data Length (DL)**															
Offset + 0x04	**Tx Data Buffer Pointer (TXDBPTR)**															
Offset + 0x06																

Figure 30-51. AAL5 TxBD

Table 30-37 describes AAL5 TxBD fields.

Table 30-37. 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 CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4	**L**	Last in frame. Set by the user to indicate the last buffer in a frame. 0 Buffer is not last in a frame 1 Buffer is last in a frame
	5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of CM.
	7–9	—	Reserved, should be cleared.
	10	**CLP**	The ATM cell header CLP bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	11	**CNG**	The ATM cell header CNG bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	12–15	—	Reserved, should be cleared.
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer, which may or may not be 8-byte-aligned. The buffer may reside in either internal or external memory. This value is not modified by the CP.

30.10.5.10 AAL1 TxBDs

Figure 30-52 shows the AAL1 TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	**R**	—	**W**	**I**	—		**CM**	—								
Offset + 0x02	**Data Length (DL)**															
Offset + 0x04	**Tx Data Buffer Pointer (TXDBPTR)**															
Offset + 0x06																

Figure 30-52. AAL1 TxBD

Table 30-38 describes AAL1 TxBD fields.

Table 30-38. AAL1 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel’s TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–11	—	Reserved, should be cleared.
0x02	—	**DL**	The number of octets the ATM controller should transmit from this BD’s buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. The buffer may reside in either internal or external memory. This value is not modified by the CP.

30.10.5.11 AAL0 TxBDs

Figure 30-53 shows AAL0 TxBDs. Note that the data length field is calculated internally as 52 bytes, plus the extra header length (defined in FPSMR[TEHS]) when in UDC mode.

	0	1	2	3	4	5	6	7	10	11	12	15
Offset + 0x00	R	—	W	I	—		CM	—		OAM	—	
Offset + 0x02	—											
Offset + 0x04	Tx Data Buffer Pointer (TXDBPTR)											
Offset + 0x06												

Figure 30-53. AAL0 TxBDs

Table 30-39 describes AAL0 TxBD fields.

Table 30-39. 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 CP clears R after the buffer has been sent or after an error occurs. 1 The buffer that the user prepared for transmission has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined by the W bit. The current table is constrained to 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–10	—	Reserved, should be cleared.
	11	**OAM**	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an F5 or F4 OAM cell. Performance monitoring calculations are not done on OAM cells.
	11–15	—	Reserved, should be cleared.

Table 30-39. AAL0 TxBD Field Descriptions

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 CP.

30.10.5.12 AAL2 TxBDs

Refer to Section 31.3.5.5, “SSSAR Transmit Buffer Descriptor.”

30.10.5.13 AAL5, AAL1 User-Defined Cell—TxBD Extension

In user-defined cell mode, the AAL5 and AAL1 TxBDs are extended to 32 bytes; see Figure 30-54.

NOTE

> For AAL0 a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

Offset + 0x08	Extra Cell Header Used to store the user-defined cell’s extra cell header. The extra cell header can be 1–12 bytes long.
Offset + 0x14	Reserved (12 Bytes)

Figure 30-54. User-Defined Cell—TxBD Extension

30.10.6 AAL1 Sequence Number (SN) Protection Table

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in dual-port RAM and is used for AAL1 only. The table should be initialized according to Figure 30-55.

	0 15
Offset + 0x00	0x0000
Offset + 0x02	0x0007
Offset + 0x04	0x000D
Offset + 0x06	0x000A
Offset + 0x08	0x000E
Offset + 0x0A	0x0009
Offset + 0x0C	0x0003
Offset + 0x0E	0x0004
Offset + 0x10	0x000B
Offset + 0x12	0x000C
Offset + 0x14	0x0006
Offset + 0x16	0x0001
Offset + 0x18	0x0005
Offset + 0x1A	0x0002
Offset + 0x1C	0x0008
Offset + 0x1E	0x000F

Figure 30-55. AAL1 Sequence Number (SN) Protection Table

30.10.7 UNI Statistics Table

The UNI statistics table, shown in Table 30-40, resides in the dual-port RAM and holds UNI statistics parameters. UNI_STATT_BASE points to the base address of this table. Each PHY's own table has a starting address given by UNI_STATT_BASE+ PHY# × 8.

Table 30-40. UNI Statistics Table

Offset[1]	Name	Width	Description
0x00	UTOPIAE	Hword	Counts cells dropped as a result of UTOPIA/ATM protocol violations. Violations include the following: 1. Parity error 2. HEC error 3. Invalid timing of RxSOC If RxClav is asserted for the selected PHY, RxSOC should be asserted the cycle immediately following the assertion of $\overline{\text{RXENB}}$. A violation occurs if RxSOC is not asserted at that time (that is, late or is missing).
0x02	MIC_COUNT	Hword	Counts misinserted cells dropped as a result of address look-up failure
0x04	CRC10E_COUNT	Hword	Counts cells dropped as a result of CRC10 failure. AAL5-ABR only
0x06	—	Hword	Reserved, should be cleared.

[1] Offset from UNI_STATT_BASE+PHY# × 8.

30.11 ATM Exceptions

The ATM controller interrupt handling involves two principal data structures: FCCEs (FCC event registers) and circular interrupt queues.

Four priority interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ], the user determines which queue receives the interrupt. Channel Rx buffer, Rx frame, or Tx buffer events can be masked by clearing interrupt mask bits in RCT and TCT.

After an interrupt request, the host reads FCCE. If FCCE[GINT*x*] = 1, at least one entry was added to one of the interrupt queues. After clearing FCCE[GINT*x*], the host processes the valid interrupt queue entries and clears each entry's valid bit. The host follows this procedure until it reaches an entry with V = 0. See Section 30.11.2, "Interrupt Queue Entry."

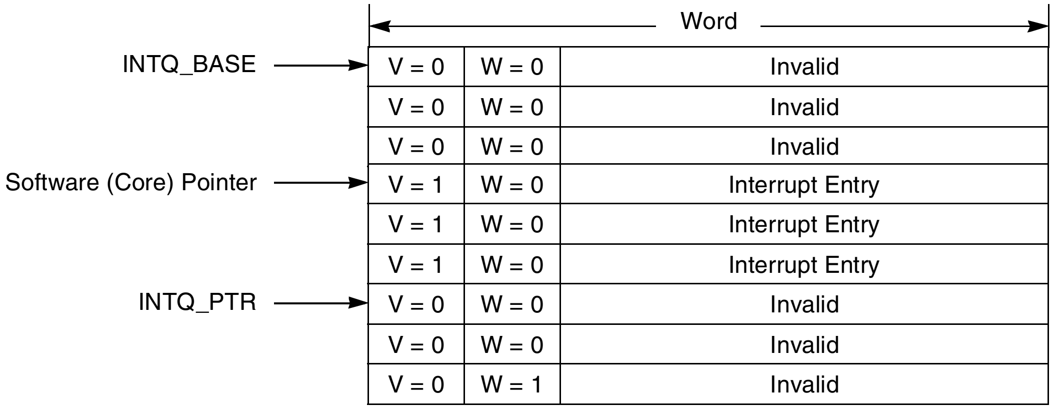
The host controls the number of interrupts sent to the core using a down counter in the interrupt queue's parameter table; see Section 30.11.3. For each event sent to an interrupt queue, a counter (that has been initialized to a threshold number of interrupts) is decremented. When the counter reaches zero, the global interrupt, FCCE[GINT*x*], is set.

30.11.1 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in a table. See Section 30.11.3, "Interrupt Queue Parameter Tables."

When an interrupt occurs, the CP writes a new entry to the interrupt queue, the V bit is set, and the queue pointer (INTQ_PTR) is incremented. Once the CP uses an entry with W = 1, it returns to the first entry in the queue. If the CP tries to overwrite a valid entry (V = 1), an overflow condition occurs and the queue's overflow flag, FCCE[INTO*x*], is set.

The interrupt queue structure is displayed in Figure 30-56.



V = 0	W = 0	Invalid
V = 0	W = 0	Invalid
V = 0	W = 0	Invalid
V = 1	W = 0	Interrupt Entry
V = 1	W = 0	Interrupt Entry
V = 1	W = 0	Interrupt Entry
V = 0	W = 0	Invalid
V = 0	W = 0	Invalid
V = 0	W = 1	Invalid

Figure 30-56. Interrupt Queue Structure

30.11.2 Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 30-57 shows an entry.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	V	—	W	—								TBNR	RXF	BSY	TXB	RXB
Offset + 0x02	Channel Code (CC)															

Figure 30-57. Interrupt Queue Entry

Table 30-41 describes interrupt queue entry fields.

Table 30-41. 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 CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	Reserved, should be cleared.
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	—	Reserved, should be cleared.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt is issued when the CP tries to open a TxBD that is not ready (R = 0). This interrupt is sent only if TCT[BNM] = 1. This interrupt has an associated channel code. Note that for AAL5, this interrupt is sent only if frame transmission is started. In this case, an abort frame transmission is sent (last cell with length=0), the channel is taken out of the APC, and the TCT[VCON] flag is cleared.
	12	RXF	Rx frame. RXF is set when an Rx frame interrupt is issued. This interrupt is issued at the end of AAL5 PDU reception. This interrupt is issued only if RCT[RXFM] = 1. This interrupt has an associated channel code.
	13	BSY	Busy condition. The BD table or the free buffer pool associated with this channel is busy. Cells were discarded due to this condition. This interrupt has an associated channel code.
	14	TXB	Tx buffer. TXB is set when a transmit buffer interrupt is issued. This interrupt is enabled when both TxBD[I] and TCT[IMK] = 1. This interrupt has an associated channel code.
	15	RXB	Rx buffer. RXB is set when an Rx buffer interrupt is issued. This interrupt is enabled when both RxBD[I] and RCT[RXBM] = 1. This interrupt has an associated channel code.
0x02	—	CC	Channel code specifies the channel associated with this interrupt

30.11.3 Interrupt Queue Parameter Tables

The interrupt queue parameters are held in parameter tables in the dual-port RAM; see Table 30-42. INTT_BASE in the parameter RAM points to the base address of these tables. Each of the four interrupt queues has its own parameter table with a starting address given by INTT_BASE+ RCT/TCT[INTQ] × 16.

Table 30-42. Interrupt Queue Parameter Table

Offset [1]	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	**INT_CNT**	Half word	Interrupt counter. Initialize with INT_ICNT. The CP decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag FCCE[GINT*x*] is set.
0x0A	**INT_ICNT**	Half word	Interrupt initial count. User-defined global interrupt threshold—the number of interrupts required before the CP issues a global interrupt (FCCE[GINT*x*]).
0x0C	**INTQ_ENTRY**	Word	Interrupt queue entry. Must be initialized to the entry pointed to by INTQ_PTR, which is initially the first empty entry of the queue. Note that after an overrun occurs, this entry must be reset to the entry pointed to by INTQ_PTR to reenable interrupt processing.

[1] Offset from INTT_BASE+RCT/TCT[INTQ] × 16.

30.12 The UTOPIA Interface

The ATM controller interfaces with a PHY device through the UTOPIA interface. The MPC8272 supports UTOPIA level 2 for both master and slave modes.

30.12.1 Extended Number of PHYs

The MPC8272 has additional pin muxing to support 31 PHYs on FCC1. To utilize this feature, do the following:

- Program CMXUAR[MAD4] = 1
- Program CMXUAR[MAD3] = 1

Refer to Chapter 37, "Parallel I/O Ports," and Section 15.4.1, "CMX Utopia Register (CMXATMR) (MPC8272 only)."

30.12.2 UTOPIA Interface Master Mode

Cell transfer on an ATM device (with single or multiple PHYs) uses cell-level handshaking as defined in the UTOPIA standards. The FCC does not pause cell transmission by the PHY and does not stop receiving cells from the PHY.

UTOPIA master signals are shown in Figure 30-58.

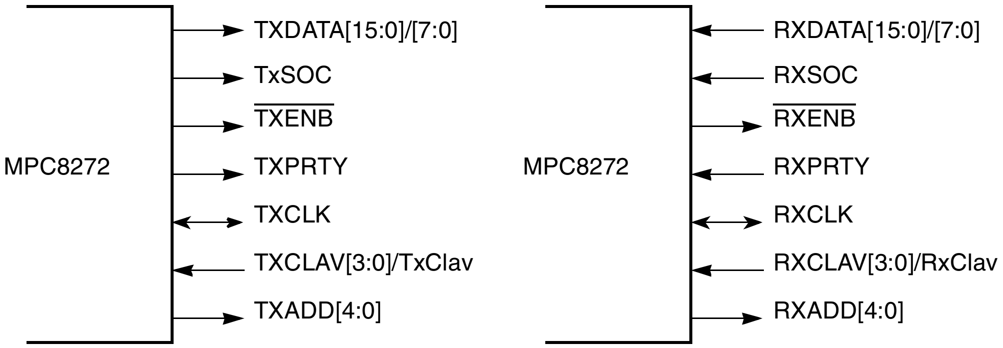


Figure 30-58. UTOPIA Master Mode Signals

Table 30-43 describes UTOPIA master mode signals.

Table 30-43. UTOPIA Master Mode Signal Descriptions

Signal	Description
TxDATA[15:0]/[7:0]	Carries transmit data from the ATM controller to a PHY device. TxDATA[15]/[7] is the msb when using UTOPIA 16/8, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by the ATM controller when the first byte of a cell is sent on TxDATA lines.
$\overline{\text{TxENB}}$	Transmit enable. Asserted by the ATM controller when valid data is placed on the TxDATA lines.
TxClav/TxCLAV[3:0]	Transmit cell available. Asserted by the PHY device to indicate that the PHY has room for a complete cell.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA bits.
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, TxPRTY signals. All the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[4:0]	Transmit address. Address bus from the ATM controller to the PHY device used to select the appropriate multi-PHY device. Each multi-PHY device needs to maintain its address. TxADD[4] is the msb.
RxDATA[15:0]/[7:0]	Carries receive data from the PHY to the ATM controller. RxDATA[15]/[7] is the msb when using UTOPIA 16/8, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the PHY device as the first byte of a cell is received on RxDATA.
$\overline{\text{RxENB}}$	Receive enable. An ATM controller asserts to indicate that RxDATA and RxSOC will be sampled at the end of the next RxCLK cycle. For multiple PHYs, $\overline{\text{RxENB}}$ is used to three-state RxDATA and RxSOC at each PHY's output. RxDATA and RxSOC should be enabled only in cycles after those with $\overline{\text{RxENB}}$ asserted.
RxClav/RxCLAV[3:0]	Receive cell available. Asserted by a PHY device when it has a complete cell to give the ATM controller.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is cleared, the cell is discarded. See Section 30.13.3, "FCC Protocol-Specific Mode Register (FPSMR)," and Section 30.10.7, "UNI Statistics Table."

Table 30-43. UTOPIA Master Mode Signal Descriptions (continued)

Signal	Description
RxCLK	Receiver clock. Synchronization reference for RxDATA, RxSOC, RxENB, RxCLAV, and RxPRTY, all of which are sampled at low-to-high transitions of RxCLK.
RxADD[4:0]	Receive address. Address bus from the ATM controller to the PHY device used to select the appropriate multi-PHY device. Each multi-PHY device needs to maintain its address. RxADD[4] is the msb.

30.12.2.1 UTOPIA Master Multiple-PHY Operation

The MPC8272 supports two polling modes:

- Direct polling uses CLAV[3:0] with PHY selection using ADD[1:0]. Up to four PHYs can be supported.
- Single-CLAV polling uses Clav and ADD[4:0]. ATM controller polls all active PHYs starting from PHY address 0x0 to the address written in FPSMR[LAST_PHY]. Up to 31 PHY devices are supported.

Both modes support round-robin priority or fixed priority, described in Section 30.13.3, "FCC Protocol-Specific Mode Register (FPSMR)."

30.12.3 UTOPIA Interface Slave Mode

In UTOPIA slave mode (single or multiple PHY), cells are transferred using cell-level and octet-level handshakes as defined by the UTOPIA level-2 standard. The FCC allows cell transfer to be halted or paused. If the master negates TXENB, the cell that the FCC is transmitting is halted. If the master negates RXENB, the cell that the FCC is receiving is paused. Note the following restriction on halting a cell transfer: there cannot be a halt immediately before the transfer of the last data word. There is no restriction on pausing a cell transfer.

UTOPIA slave signals are shown in Figure 30-59.

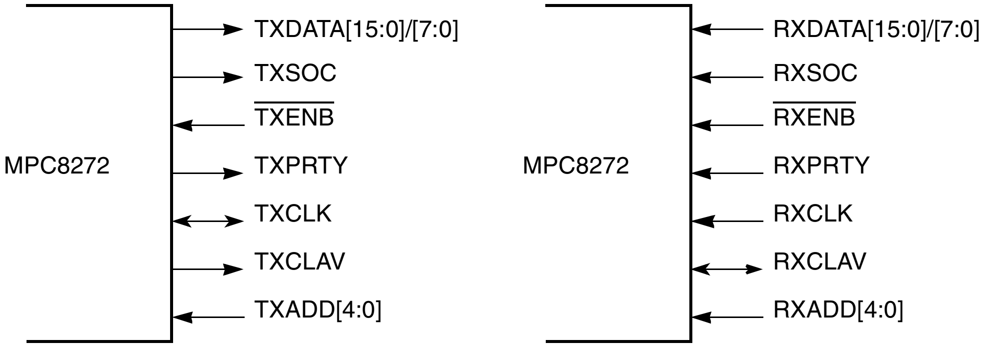


Figure 30-59. UTOPIA Slave Mode Signals

Table 30-44 describes UTOPIA slave mode signals.

Table 30-44. UTOPIA Slave Mode Signals

Signal	Description
TxDATA[15:0]/[7:0]	Transmit data bus. Carries transmit data from the ATM controller to the master device. TxDATA[15]/[7] is the msb, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by an ATM controller as the first byte of a cell is sent on the TxDATA lines.
$\overline{\text{TxENB}}$	Transmit enable. An input to the ATM controller. It is asserted by the UTOPIA master to signal the slave to send data in the next TxCLK cycle.
TxCLAV	Transmit cell available. Asserted by the ATM controller to indicate it has a complete cell to transmit.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA.
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, and TxPRTY signals. All of the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[4:0]	Transmit address. Address bus from the master to the ATM controller used to select the appropriate mulit-PHY device.
RxDATA[15:0]/[7:0]	Receive data bus. Carries receive data from the master to the ATM controller. RxDATA[15]/[7] is the msb, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the master device whenever the first byte of a cell is being received on the RxDATA lines.
$\overline{\text{RxENB}}$	Receive enable. Asserted by the master device to signal the slave to sample the RxDATA and RxSOC signals.
RxCLAV	Receive cell available. Asserted by the ATM controller to indicate it can receive a complete cell.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA[15–0]. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is cleared, the cell is discarded. See Section 30.13.3, "FCC Protocol-Specific Mode Register (FPSMR)," and Section 30.10.7, "UNI Statistics Table."
RxCLK	Receive clock. Provides the synchronization reference for the RxDATA, RxSOC, $\overline{\text{RxENB}}$, RxCLAV, and RxPRTY signals. All the above signals are sampled at low-to-high transitions of RxCLK.
RxADD[4:0]	Receive address. Address bus from master to the ATM controller device used to select the appropriate multi-PHY device.

30.12.3.1 UTOPIA Slave Multiple-PHY Operation

The user should write the ATM controller PHY address in FPSMR[PHY ID].

30.12.3.2 UTOPIA Clocking Modes

The UTOPIA clock can be generated internally or externally. If the UTOPIA clock is to be generated internally, the user should assign one of the baud-rate generators to supply the UTOPIA clock. See Chapter 15, "CPM Multiplexing."

30.12.3.3 UTOPIA Loop-Back Modes

The UTOPIA interface supports loop-back mode. In this mode, the Rx and Tx UTOPIA signals are shorted internally. Output pins are driven; input pins are ignored.

Note that in loop-back mode, the transmitter and receiver must operate in complementary modes. For example, if the transmitter is master, the receiver must be a slave (FPSMR[TUMS] = 0, FPSMR[RUMS] = 1).

Modes are selected through GFMR[DIAG], as shown in Table 30-45.

Table 30-45. UTOPIA Loop-Back Modes

DIAG	Description
00	Normal mode
01	Loop-back. UTOPIA Rx and Tx signals are shorted internally. Output pins are driven, input pins are ignored.
1x	Reserved

30.13 ATM Registers

The following sections describe the configuration of the registers in ATM mode.

30.13.1 General FCC Mode Register (GFMR)

The GFMR mode field should be programmed for ATM mode. To enable transmit and receive functions, ENT and ENR must be set as the last step in the initialization process. Full GFMR details are given in Section 29.2, “Mode Registers.”

30.13.2 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3 7
Field	TIREM	LPB	CLK	—
Reset	0000_0000			
R/W	R/W			
Addr	0x11390 (GFEMR1), 0x113B0(GFEMR2)			

Figure 30-60. General FCC Expansion Mode Register (GFEMR)

Table 30-46 describes GFEMR*x* fields.

Table 30-46. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode. Internal rate for PHY[0:3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode. PHY[0:31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI and FIRSR_LO. Underrun status for PHY[0:31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode, mixing of internal rate and external rate is not enabled.
1	LPB	RMII loopback diagnostic mode (Ethernet mode) 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50-MHz input clock from external oscillator (Ethernet mode) 0 50 MHz (for fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared.

30.13.3 FCC Protocol-Specific Mode Register (FPSMR)

The FCC protocol-specific mode register (FPSMR), shown in Figure 30-61, controls various protocol-specific FCC functions. The user should initialize the FPSMR. Erratic behavior may result if there is an attempt to write to the FPSMR while the transmitter and receiver are enabled.

	0 3	4 7	8	9	10	11 15
Field	TEHS	REHS	ICD	TUMS	RUMS	LAST PHY/PHY ID
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x11304 (FPSMR1), 0x11324 (FPSMR2)					

	16 17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—	TPRI	TUDC	RUDC	RXP	TUMP	—	TSIZE	RSIZE	UPRM	UPLM	RUMP	HECI	HECC	COS
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x11306 (FPSMR1), 0x11326 (FPSMR2)														

Figure 30-61. FCC ATM Mode Register (FPSMR)

Table 30-47 describes FPSMR fields.

Table 30-47. FCC ATM Mode Register (FPSMR)

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	Idle cells discard 0 Discard idle cells (GFC, VPI, VCI, PTI =0). 1 Do not discard idle cells.
9	TUMS	Transmit UTOPIA master/slave mode 0 Transmit UTOPIA master mode is selected. 1 Transmit UTOPIA slave mode is selected.
10	RUMS	Receive UTOPIA master/slave mode 0 Receive UTOPIA master mode is selected. 1 Receive UTOPIA slave mode is selected.
11–15	LAST PHY/ PHY ID	Last PHY. (Multi-PHY master mode only.) The UTOPIA interface polls all PHYs starting from PHY address 0 and ending with the PHY address specified in LAST PHY. (The number of active PHYs are LAST PHY+1.) LAST PHY should be specified in both single-Clav and direct-polling modes. PHY ID. (Multi-PHY slave mode only.) Determines the PHY address of the ATM controller when configured as a slave in a multiple PHY ATM port. **Note:**
16–17	—	Reserved, should be cleared.
18	TPRI	Transmitter priority. Used to adjust the default priority of the FCC transmitter. It is strongly recommended to set TPRI when in multi-PHY mode, or in single-PHY mode if the maximal bit rate (either internal or external rate) is higher than that of the other FCCs; for other modes, it should remain cleared. 0 Default operation 1 Prevents elevation to emergency mode Refer to Table 13-2.
19	TUDC	Transmit user-defined cells 0 Regular 53-byte cells 1 User-defined cells
20	RUDC	Receive user-defined cells 0 Regular 53-byte cells 1 User-defined cells
21	RxP	Receive parity check 0 Check Rx parity line 1 Do not check Rx parity line
22	TUMP	Transmit UTOPIA multiple-PHY mode 0 Transmit UTOPIA single-PHY mode is selected 1 Transmit UTOPIA multiple-PHY mode is selected
23–25	—	Reserved, should be cleared.

Table 30-47. FCC ATM Mode Register (FPSMR)

Bits	Name	Description
26	UPRM	UTOPIA priority mode 0 Round robin. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll the next PHY in order. 1 Fixed priority. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll from PHY zero.
27	UPLM	UTOPIA polling mode. 0 Single Clav polling. Polling is done using Add[4:0] and Clav. Selection is done using Add[4:0]. Up to 31 PHYs can be polled. 1 Direct polling. Polling is done using Clav[3:0]. Selection is done using Add[1:0]. Up to 4 PHYs can be polled.
28	RUMP	Receive UTOPIA multiple-PHY mode. 0 Receive UTOPIA single-PHY mode is selected 1 Receive UTOPIA multiple-PHY mode is selected
29	HECI	HEC included. Used in UDC mode only. 0 HEC octet is not included when UDC mode is enabled. 1 HEC octet is included when UDC mode is enabled.
30	HECC	Receive HEC check 0 Do not check Rx HEC. 1 Check Rx HEC. HEC errors are reported in UTOPIAE counter. (See Section 30.10.7, “UNI Statistics Table.”) This option can be used only in UTOPIA 8-bit data bus size mode.
31	COS	Coset mode enable 0 Check Rx HEC with no COSET 1 Check Rx HEC with COSET mode enabled

30.13.4 ATM Event Register (FCCE)/Mask Register (FCCM)

The FCCE register is the ATM controller event register when the FCC operates in ATM mode. When it recognizes an event, the ATM controller sets the corresponding FCCE bit. Interrupts generated by this register can be masked in FCCM. FCCE is memory-mapped and can be read at any time. Bits are cleared by writing ones to them; writing zeros has no effect. Unmasked bits must be cleared before the CP clears the internal interrupt request.

FCCM is the ATM controller mask register. The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

	0–4	5	6	7	8	9	10	11	12	13	14	15
Field	—	TIRU	GRLI	GBPB	GINT3	GINT2	GINT1	GINT0	INTO3	INTO2	INTO1	INTO0
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x11310 (FCCE1), 0x11330 (FCCE2), 0x11314 (FCCM1), 0x11334 (FCCM2)											

	16–31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x11312 (FCCE1), 0x11332 (FCCE2), 0x11316 (FCCM1), 0x11336 (FCCM2)

Figure 30-62. ATM Event Register (FCCE)/FCC Mask Register (FCCM)

Table 30-48 describes FCCE fields.

Table 30-48. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	TIRU	Transmit internal rate underrun. A cumulative lag of seven cells has formed between the programmable rate and the actual rate for a specific Phy. A transmit internal rate counter expired and a cell was not sent, either because of slow CPM performance or slow PHY performance. TIRU may be set only when using transmit internal rate mode; see Section 30.13.5, “FCC Transmit Internal Rate Registers (FTIRR1).”
6	GRLI	Global red-line interrupt. GRLI is set when a free buffer pool’s RLI flag is set. The RLI flag is also set in the free buffer pool’s parameter table.
7	GBPB	Global buffer pool busy interrupt. GBPB is set when a free buffer pool’s BUSY flag is set. The BUSY flag is also set in the free buffer pool’s parameter table.
8–11	GINT*x*	Global interrupt. Set when the number of events sent to the corresponding interrupt queue reaches the corresponding event threshold. See Section 30.11, “ATM Exceptions.”
12–15	INTO*x*	Interrupt queue overflow. Set when an overflow condition occurs in the corresponding interrupt queue. This occurs when the CP attempts to overwrite a valid interrupt entry. See Section 30.11.1, “Interrupt Queues.”
16–31	—	Reserved, should be cleared.

30.13.5 FCC Transmit Internal Rate Registers (FTIRR1)

NOTE: Internal Rate Source Clock

> The source clock for internal rate is configured in CMXATMR register. The frequency of this clock must be less then one third of the serial transmit clock frequency.

The first four PHY devices (address 00–03) on FCC1 have their own transmit internal rate registers (FTIRR1_PHY0–FTIRR1_PHY3) for use in transmit internal rate mode. In this mode, the total transmission rate is determined by FCC internal rate timers. As a master, the controller only polls the PHY’s Clav status at the rate determined by the internal rate. As a slave, the controller attempts to insert cells into the FIFO at the internal rate. The controller can handle a lag of up to 7 cells per PHY between the programmable and actual bus rate. When the cell count mismatch reaches seven, a TIRU event is reported; see Section 30.13.4, “ATM Event Register (FCCE)/Mask Register (FCCM).” Note that a mismatch occurs if the PHY rate or the CPM performance is lower than the internal rate. FTIRR*x*, shown in Figure 30-63, includes the initial value of the internal rate timer. The source clock of the internal rate timers is supplied by one of four baud-rate generators selected in CMXUAR; see Section 15.4.1, “CMX Utopia Register (CMXATMR) (MPC8272 only).” Note that in slave mode, FTIRR1_PHY0 is used regardless of the slave PHY address.

	0	1	2	3	4	5	6	7
Field	TRM	Initial Value						
Reset	0000_0000							
R/W	R/W							
Addr	FCC1: 0x1131C (FTIRR1_PHY0), 0x1131D (FTIRR1_PHY1), 0x1131E (FTIRR1_PHY2), 0x1131F (FTIRR1_PHY3)							

Figure 30-63. FCC Transmit Internal Rate Registers (FTIRR1)

Table 30-49 describes FTIRR1 fields.

Table 30-49. FTIRR1 Field Descriptions

Bits	Name	Description
0	TRM	Transmit mode. 0 External rate mode 1 Internal rate mode
1–7	Initial Value	The initial value of the internal rate timer. A value of 0x7F produces the minimum clock rate (BRG CLK divided by 128); 0x00 produces the maximum clock rate (BRG CLK divided by 1).

Figure 30-64 shows how transmit clocks are determined.

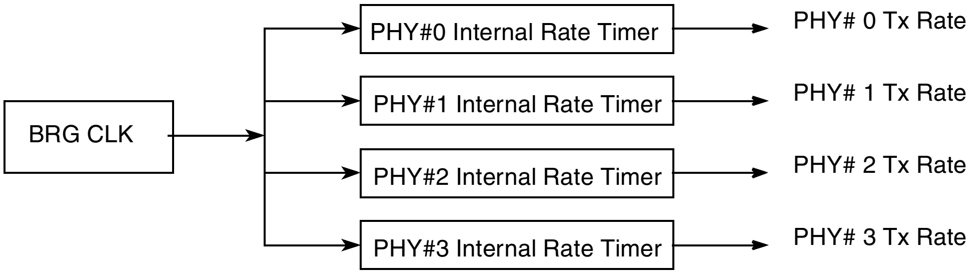

BRG CLK
PHY#0 Internal Rate Timer
PHY#1 Internal Rate Timer
PHY#2 Internal Rate Timer
PHY#3 Internal Rate Timer
PHY# 0 Tx Rate
PHY# 1 Tx Rate
PHY# 2 Tx Rate
PHY# 3 Tx Rate

Figure 30-64. FCC Transmit Internal Rate Clocking

30.13.5.1 Example

Suppose the MPC8272 is connected to four 155 Mbps PHY devices; the maximum transmission rate is 155 Mbps for the first PHY and 10 Mbps for the rest of the PHYs. The BRG CLK should be set according to the highest rate. If the CPM clock is 133 MHz and the BRG clock is 66 MHz, the BRG should be programmed to divide the CPM clock by 181 to generate cell transmit requests every 362 system clocks:

$$\frac{(66\text{MHz} \times (53 \times 8))}{155.52\text{Mbps}} = 181$$

For the 155-Mbps PHY, the FTIRR divider should be programmed to zero (the BRG CLK is divided by one); for the rest of the 10-Mbps PHYs, the FTIRR divider should be programmed to 14 (the BRG CLK is divided by 15).

See also Section 30.16.1, “Using Transmit Internal Rate Mode.”

30.14 Expanded Internal Rate

NOTE: Internal Rate Source Clock

The source clock for internal rate is configured in CMXATMR register. The frequency of this clock must be less then one third of the serial transmit clock frequency.

30.14.1 Transmit External Rate and Internal Rate Modes

The ATM controller supports the following three rate modes:

- External rate mode—The total transmission rate is determined by the PHY transmission rate. The FCC sends cells to keep the PHY FIFOs full; the FCC inserts idle/unassign cells to maintain the transmission rate.
- Internal rate mode—The total transmission rate is determined by the FCC internal rate timers. In this mode, the FCC does not insert idle/unassign cells. The internal rate mechanism is supported for the first four PHY devices (PHY addresses 0–3). Each PHY has its own FTIRR, described in Section 30.13.5, “FCC Transmit Internal Rate Registers (FTIRR1).” The FTIRR includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer expires. When using internal rate mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers.
- Internal rate expanded mode—The total transmission rate is determined by the FCC internal rate timers and by the assignment of rate per PHY. In this mode, the FCC does not insert idle/unassign cells. The internal rate expanded mode differs from the internal rate mode in that the internal rate mechanism is extended for 31 PHY devices (PHY addresses 0–30) and there cannot be a mix of external and internal rate PHYs. Expanded internal rate is configured by registers GFEMRx, FIRPERx, FIRSRx_HI, FIRSRx_LO, and by FTIRRx. Another feature of internal rate expanded mode is an indication of transmit underrun error status per PHY. When using internal rate expanded mode, the user assigns one of the BRGs to clock the four internal rate timers, and any timer can trigger any PHY.

30.14.2 FCC Transmit Internal Rate Mode

In internal rate mode the total transmission rate is the sum of the rates assigned for all PHYs. This register controls how internal rate is configured. In internal rate mode (GFEMR[TIREM] = 0), the internal rate assigned per PHY is configured by registers FTIRR[0–3]. In internal rate expanded mode (GFEMR[TIREM] = 1), registers FTIRR[0–3] control the available rates, but the PHY settings are configured in registers FIRPER, FIRSR_HI and FIRSR_LO. In TIREM = 0 mode, internal rate can only be used for PHY[0-3], whereas in TIREM = 1 mode up to 31 PHYs are supported. If TIREM = 1 mode is selected, the transmit internal rate underrun (TIRU) status per PHY may be read at any time in register FIRER.

30.14.3 FCC Transmit Internal Rate Port Enable Register (FIRPER)

This register enables internal rate transmission for PHY[0:30]. It is valid only if GFEMR[TIREM] = 1. If a PHY is not enabled in FIRPER, all TxClav indications from that PHY will be masked. The user should configure FIRPER according to the PHY addresses that are being used on the UTOPIA bus and should not enable PHYs with addresses larger than the last PHY address set by FPSMR[Last PHY]. PHYs can be enabled or disabled at any time, for example, if a TIRU event has occurred.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PE0	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	PE11	PE12	PE13	PE14	PE15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11380 (FIRPER1)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PE16	PE17	PE18	PE19	PE20	PE21	PE22	PE23	PE24	PE25	PE26	PE27	PE28	PE29	PE30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11382 (FIRPER1)															

Figure 30-65. FCC Transmit Internal Rate Port Enable Register (FIRPER)

Table 30-50 describes FIRPER1 fields.

Table 30-50. FIRPER1 Field Descriptions (TIREM=1)

Bit	Name	Description
0–15	PEy	Port enable 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_HI (refer to Section 30.14.5, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
16–30	PEy	Port enable 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_LO (refer to Section 30.14.5, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
31	—	Reserved, should be cleared.

30.14.4 FCC Internal Rate Event Register (FIRER)

Transmit internal rate underrun (TIRU) errors are reported for any PHY that has a transmission deficiency of 8 cells. Under this condition and in internal rate mode only, FCCE[TIRU] is set, and if the corresponding bit in the FCC mask register (FCCM[TIRU]) is set, an interrupt is generated. If TIREM = 1, the TIRU status per PHY can be read at any time in the FCC internal rate event register (FIRER). Once FIRER[TIRUy] error status is set, it can be cleared only by writing 1 to it. To prevent an underrun PHY from continuously reporting errors, it can be disabled by FIRPER. The sequence of disabling a PHY is as follows:

- Disable PHY y by clearing FIRPER[y].
- Clear event FIRER[y] by writing 1 to it.
- Clear event FCCE[TIRU] by writing 1 to it.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TIRU 0	TIRU 1	TIRU 2	TIRU 3	TIRU 4	TIRU 5	TIRU 6	TIRU 7	TIRU 8	TIRU 9	TIRU 10	TIRU 11	TIRU 12	TIRU 13	TIRU 14	TIRU 15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11384 (FIRER1)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TIRU 16	TIRU 17	TIRU 18	TIRU 19	TIRU 20	TIRU 21	TIRU 22	TIRU 23	TIRU 24	TIRU 25	TIRU 26	TIRU 27	TIRU 28	TIRU 29	TIRU 30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11386 (FIRER1)															

Figure 30-66. FCC Internal Rate Event Register (FIRER)

Table 30-51 describes FIRER1 fields.

Table 30-51. FIRER1 Field Descriptions (TIREM=1)

Bit	Name	Description
0–30	TIRUy	Transmit internal rate underrun 0 There is no transmission underrun for this PHY. 1 Transmit internal rate underrun or PHY address y has occurred. Bit is cleared by writing 1 to it. Writing 0 has no effect on value.
31	—	Reserved, should be cleared.

30.14.5 FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)

If TIREM = 1, each PHY can be assigned one of four rates, as configured by the four FCC transmit internal rate timers. The FCC internal rate selection registers (FIRSRx_HI, FIRSRx_LO), shown in Figure 30-67 and Figure 30-68, assign rate group to each of the PHYs.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	GS0		GS1		GS2		GS3		GS4		GS5		GS6		GS7	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11388 (FIRSR1_HI)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	GS8		GS9		GS10		GS11		GS12		GS13		GS14		GS15	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x1138A (FIRSR1_HI)															

Figure 30-67. FCC Internal Rate Selection Register HI (FIRSR1_HI)

Table 30-52 describes FIRSR1_HI fields.

Table 30-52. IRSR1_HI Field Descriptions (TIREM=1)

Bit	Name	Description
0–31	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP3.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	GS16		GS17		GS18		GS19		GS20		GS21		GS22		GS23	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x1138C (FIRSR1_LO)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	GS24		GS25		GS26		GS27		GS28		GS29		GS30		—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x1138E (FIRSR1_LO)															

Figure 30-68. FCC Internal Rate Selection Register LO (FIRSR1_LO)

Table 30-53 describes FIRSR1_LO fields.

Table 30-53. FIRSR1_LO Field Descriptions (TIREM = 1)

Bit	Name	Description
0–29	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP3.
30–31	—	Reserved, should be cleared.

30.14.6 FCC Transmit Internal Rate Register (FTIRR1)

If GFEMR[TIREM] = 0, PHYs at addresses 0–3 have their own FCC transmit internal rate registers (FTIRR1_PHY0–FTIRR1_PHY3) for use in transmit internal rate mode. If TIREM=1, FTIRRx are used as group timers and PHYs at addresses 0-30 are assigned to a rate group by FIRSR1_HI and FIRSR1_LO. FTIRR1, shown in Figure 30-63, includes the initial value of the internal rate timer. The clock to the internal rate timers is supplied by one of four baud-rate generators selected in CMXUAR; refer to Section 15.4.1, “CMX Utopia Register (CMXATMR) (MPC8272 only).” Note that in slave mode, FTIRR0 is used regardless of the slave PHY address.

	0	1	7
Field	TRM	Initial Value	
Reset	0000_0000		
R/W	R/W		
Address	GFEMR[TIREM=0] FCC1: 0x1131C (FTIRR1_PHY0), FCC1: 0x1131D (FTIRR1_PHY1), FCC1: 0x1131E (FTIRR1_PHY2), FCC1: 0x1131F (FTIRR1_PHY3),	GFEMR[TIREM=1] FCC1: 0x1131C (FTIRR1_GRP0), FCC1: 0x1131D (FTIRR1_GRP1), FCC1: 0x1131E (FTIRR1_GRP2), FCC1: 0x1131F (FTIRR1_GRP3),	

Figure 30-69. FCC Transmit Internal Rate Register (FTIRR)

Table 30-49 describes FTIRR1 fields.

Table 30-54. FTIRR1 Field Descriptions

Bit	Name	Description
0	TRM (TIREM=0)	PHY transmit mode 0 External rate mode 1 Internal rate mode
	TRM (TIREM=1)	Group transmit mode 0 Group rate timer [x] disabled 1 Internal rate timer for Group[x] is enabled and division factor is set by Initial Value field.
1–7	Initial value	The initial value of the internal rate timer. A value of 0x7F produces the minimum clock rate (BRG CLK divided by 128); 0x00 produces the maximum clock rate (BRG CLK divided by 1).

Figure 30-64 shows how transmit clocks are determined.

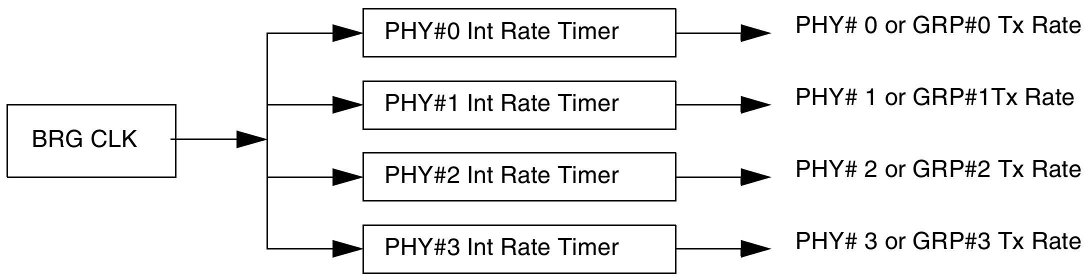


Figure 30-70. FCC Transmit Internal Rate Clocking

30.14.6.1 Example

If the MPC8272 is connected to four 155 Mbps PHY devices; the maximum transmission rate is 155 Mbps for the first PHY and 10 Mbps for the rest of the PHYs, the BRG CLK should be set according to the highest rate. If the system clock is 133 MHz, the BRG should be programmed to divide the system clock by 362 to generate cell transmit requests every 362 system clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{155.52\text{Mbps}} = 362$$

For the 155 Mbps PHY, the FTIRR divider should be programmed to zero (the BRG CLK is divided by one); for the rest of the 10 Mbps PHYs, the FTIRR divider should be programmed to 14 (the BRG CLK is divided by 15).

30.14.7 Internal Rate Programming Model

The programming sequence in TIREM = 0 mode is as follows:

1. Clear GFEMRx[TIREM].
2. Program FTIRR1.

The programming sequence in TIREM = 1 mode is as follows:

1. Clear FTIRR1[TRM].
2. Set GFEMRx[TIREM].
3. Program FIRSR1_HI and FIRSR1_LO.
4. Program FTIRR1.
5. Program FIRPER1.

If FTIRR1 is set to generate same order of magnitude rates, setting round-robin polling mode is more adequate than fixed-priority mode. To reduce the risk of transmit underrun if there are a few PHYs with high internal rate and a number of PHYs with a low internal rate, the fast PHYs should be assigned consecutive addresses starting at 0, and fixed-priority mode should be chosen.

30.15 ATM Transmit Command

The CPM command set includes an ATM TRANSMIT that can be sent to the CP command register (CPCR), described in Section 13.4.1.

The ATM TRANSMIT command (CPCR[opcode] = 0b1010, CPCR[SBC[code]] = 0b01110, CPCR[SBC[page]] = 0b00100 or 0b00101, CPCR[MCN] = 0b0000_1010) turns a passive channel into an active channel by inserting it into the APC scheduling table. Note that an ATM TRANSMIT command should be issued only after the channel's TCT is completely initialized and the channel has BDs ready to transmit. Note also that CPCR[SBC[code]] = 0b01110 and not FCC1 code.

Before issuing the command, the user should initialize COMM_INFO fields in the parameter RAM as described in Figure 30-71.

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x86	—					CTB	PHY#					ACT		PRI		
0x88	Channel Code (CC)															
0x8A	BT															

Figure 30-71. COMM_INFO Field

Table 30-55 describes COMM_INFO fields.

Table 30-55. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x86	0–4	—	Reserved, should be cleared.
	5	CTB	Connection tables bus. Used for external channels only 0 External connection tables reside on the 60x bus. 1 Reserved.
	6–10	PHY#	PHY number. In single PHY mode this field should be cleared In multiple PHY mode this field is an index to the APC parameter table associated with this channel.
	11–12	ACT	ATM channel type 00 Other channel 01 VBR channel 1x Reserved
	13–15	PRI	APC priority level. 000 Highest priority (APC_LEVEL1) 111 Lowest priority (APC_LEVEL8).
0x88	0–15	CC	Channel code. The channel code associated with the current channel.
0x8A	0–15	BT	Burst tolerance. For use by VBR channels only (ACT field is 0b01). Specifies the initial burst tolerance (GCRA burst credit) of the current VC.

30.16 Configuring the ATM Controller for Maximum CPM Performance

The following sections recommend ATM controller configurations to maximize CPM performance.

30.16.1 Using Transmit Internal Rate Mode

When the total transmit rate is less than the PHY rate, use the transmit internal rate mode and configure the internal rate clock to the maximum bit rate required. (See 30.2.1.5, "Transmit External Rate and Internal Rate Modes.") The PHY, not the ATM controller, then automatically fills the unused bandwidth with idle cells. If the internal rate mode is not used, CPM performance is consumed, generating the idle cell payload and using the scheduling algorithm to fill the unused bandwidth at the higher PHY rate.

For example, suppose a system uses a 155.52-Mbps OC-3 device as PHY0, but the maximum required data rate is only 100 Mbps. In transmit internal rate mode, the user can configure the internal rate mechanism to clock the ATM transmitter at a cell rate of 100 Mbps. If the system clock is 133 MHz, program a BRG to divide the system clock by 563 to generate a transmit cell request every 563 CPM clocks:

$$\frac{(133\mathrm{MHz} \times (53 \times 8))}{100\mathrm{Mbps}} = 563$$

Set FTIRR*x*_PHY0[TRM] to enable the transmit internal rate mode and clear FTIRR*x*_PHY0[Initial Value] since there is no need to further divide the BRG. See Section 30.13.5, "FCC Transmit Internal Rate Registers (FTIRR1)."

In external rate mode, however, the transmit cell request frequency is determined by the PHY's maximum rate, not by internal FCC counters. If an OC-3 PHY is used with the ATM controller in external rate mode, the requests must be generated every 362 CPM clocks (assuming a 133-MHz CPM clock). If only 100 Mbps is used for real data, 36% of the transmit cell requests consume CPM processing time sending idle cells.

30.16.2 APC Configuration

Maximizing the number of cells per slot (CPS) and minimizing the priority levels defined in the APC data structure improves CPM performance:

- Cells per slot. CPS defines the maximum number of ATM cells allowed to be sent during a time slot. (See Section 30.3.3.1, "Determining the Cells Per Slot (CPS) in a Scheduling Table.") The scheduling algorithm is more efficient sending multiple cells per time slot using the linked-channel field. Therefore, choose the maximum number of cells per slot allowed by the application.
- Priority levels. The user can configure the APC data structure to have from one to eight priority levels. (See Section 30.3.6, "Determining the Priority of an ATM Channel.") For each time slot, the scheduling algorithm scans all priority levels and maintains pointers for each level. Therefore, enable only the minimum number of priority levels required.

30.16.3 Buffer Configuration

Using statically allocated buffers of optimal sizes also improves CPM performance:

- Buffer size. Opening and closing buffer descriptors consumes CPM processing time. Because smaller buffers require more opening and closing of BDs, the optimal buffer size for maximum CPM performance is equal to the packet size (an AAL5 frame, for example).
- Free buffer pool. When the free buffer pool is used, the CPM dynamically allocates buffers and links them to a channel’s BD. In static buffer allocation, the core assigns a fixed data buffer to each BD. (See Section 30.10.5.2, “Receive Buffer Operation.”) When allowed by the application, use static buffer allocation to increase CPM performance.

Chapter 31
AAL2 Protocol

NOTE

> The functionality described in this chapter is not available on the MPC8248 and the MPC8247.
>
> Refer to www.freescale.com for the latest RAM microcode packages that support enhancements.

The microcode implementation of the ATM adaptation layer type 2 (AAL2) on the MPC8272 is compliant with ITU-T recommendations I.363.2 and I.366.1. This chapter describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR and should be used as a supplement to Chapter 30, "ATM Controller and AAL0, AAL1, and AAL5 Protocols."

31.1 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 31-1). 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.

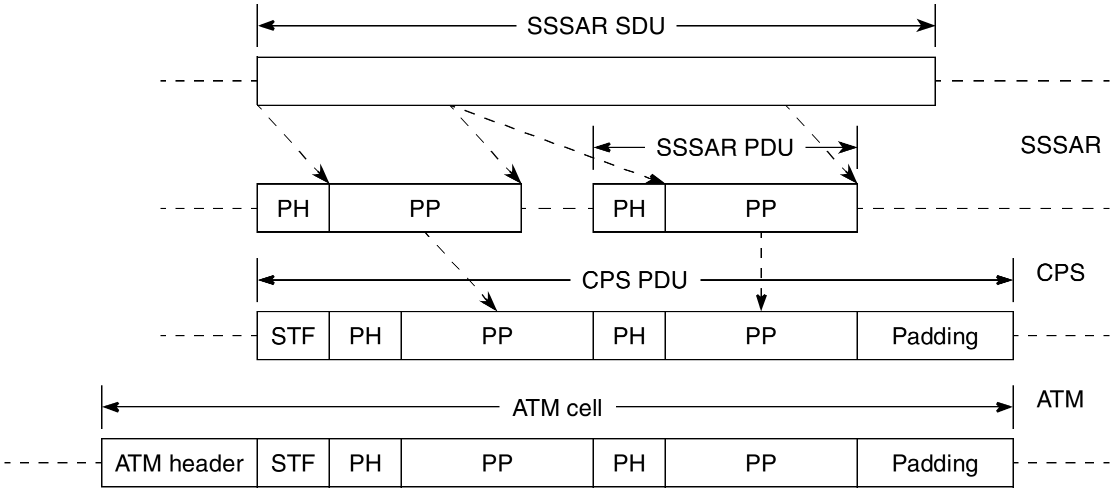


Figure 31-1. AAL2 Data Units

AAL2 is subdivided into two sublayers, as shown in Figure 31-2:

- Common part sublayer (CPS)—In the CPS sublayer, variable length packets coming from multiple users are assembled into CPS-PDUs belonging to a single ATM VC.

- Service-specific convergence sublayer (SSCS)—The SSCS sublayer handles the mapping of user data to the CPS sublayer. The SSCS segments large data frames into smaller CPS packets and also provides different services to the user, such as transmission error detection. The SSCS sublayer is further divided into three service-specific layers:
  - — Service-specific segmentation and reassembly sublayer (SSSAR)
  - — Service-specific transmission error detection sublayer (SSTED)
  - — Service-specific assured data transfer sublayer (SSADT)

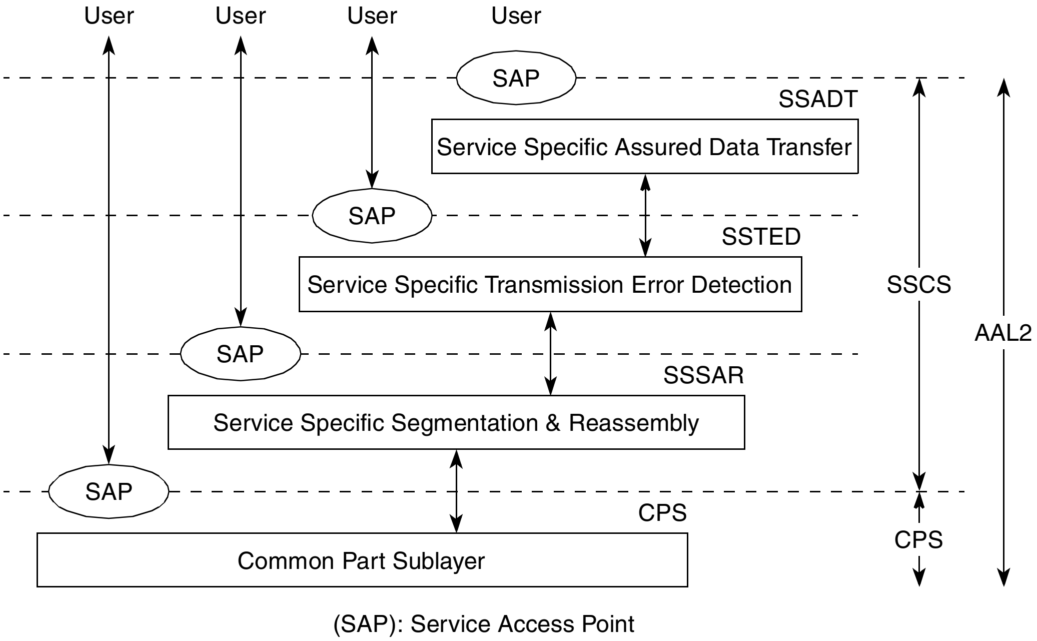

User
User
User
User
SAP
SSADT
Service Specific Assured Data Transfer
SAP
SSTED
Service Specific Transmission Error Detection
SSCS
AAL2
SAP
SSSAR
Service Specific Segmentation & Reassembly
SAP
CPS
CPS
Common Part Sublayer
(SAP): Service Access Point

Figure 31-2. AAL2 Sublayer Structure

The AAL2 microcode implements the CPS and SSSAR sublayers. (The SSADT and SSTED sublayers are not implemented.) As shown in Figure 31-2, 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 31-3.

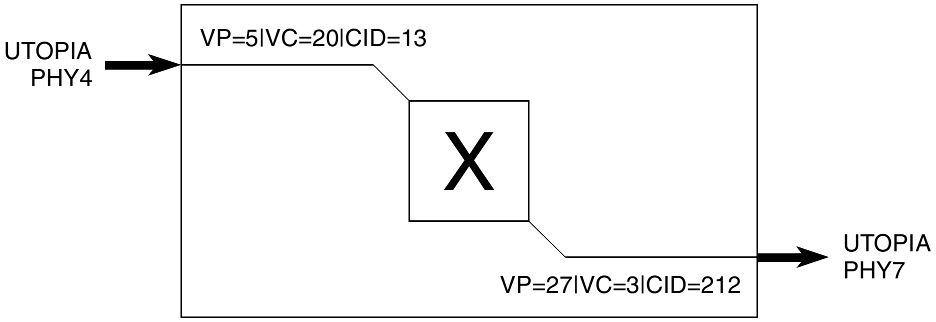


Figure 31-3. AAL2 Switching Example

31.2 Features

The MPC8272's AAL2 features are as follows:

- Fully complies with ITU-T I.363.2 (09/97 and 11/00) and ITU-T I.366.1 (06/98) specifications
- Number of AAL2 external channels supported is subject to internal memory constraints
  - Each external channel requires space for one transmit queue descriptor in internal memory. Typically, up to 1000 external channels can be supported.
- Supports CBR, VBR and UBR+ traffic types
  - PCR pacing (with optional Timer_CU)
  - VBR pacing (with optional Timer_CU)
  - UBR+ pacing (no Timer_CU support)
- Priority mechanism for transmitting per VC. The priority mechanism provides for TX queues having equal or differing priorities. The SSSAR TX queues can be prioritized flexibly among the CPS TX queues.
- Timer_CU support
- NoSTF mode support
- Support for partially filled cells
- User-defined cells (as described in Section 30.7, "User-Defined Cells (UDC)")
- Interrupt indications include the ATM channel number, the CID, and the event type. The events reported are TX buffer not ready, TX buffer transmitted, RX buffer not ready, RX buffer, RX SSSAR frame, and RX AAL2 error events.
- CPS switching
  - Switching from a receive PHY$_1$ | VP$_1$ | VC$_1$ | CID$_1$ combination to another transmit PHY$_2$ | VP$_2$ | VC$_2$ | CID$_2$ combination
  - Partial packet discard support
  - For each switched queue, a counter for the total number of packets in the queue is available.
- CPS receiver
  - Segmentation of CPS PDU directly to external memory queues
  - A separate queue for every VP | VC | CID or a common queue for multiple VP | VC | CID combinations
  - Receive one or multiple VP | VC | CIDs directly to a specific TX queue to enable switching
  - Sequence number (SN) protection check for CPS-PDU
  - CRC5 (HEC) check to detect errors in the CPS-PH of the CPS-Packet
  - OSF (offset field) of the STF (start of frame) check (a valid value is less than 48)
  - An SDU length limit parameter (Max_SDU_Deliver_Length) per ATM VC
  - Odd parity check for the STF octet of the CPS-PDU
- CPS transmitter
  - Reassemble CPS PDU directly from external memory
  - Perform CPS-PDU padding as needed

  - Insert Sequence Number bit of the CPS-PDU
  - Parity bit is calculated to provide odd parity over the STF octet
  - Calculation of CRC-5 on the first 19 bits of the CPS-PH
  - UUI field in the CPS-Packet header is programmed according to the value of the CPS_UUI parameter (per packet)
  - A free running counter (per TX Queue) is decremented for each packet sent
- SSSAR receiver
  - Reassemble CPS packets from the same CID into an SSSAR SDU
  - A separate queue for every PHY | VP | VC | CID
  - Perform all the above mentioned CPS receiver functions
  - A Ras_Timer mode is provided. When the Ras_Timer expires the buffer is closed with a timer expired error. The next packet received starts a new SSSAR SDU in a new buffer.
  - The SDU length is checked. If the frame exceeds the length limit (SSSAR_Max_SDU_Length), the receiver discards the rest of the packets from the current frame, closes the buffer and reports a Max_SDU violation error in the BD. The next packet received starts a new SSSAR SDU in a new buffer.
  - Partial packet discard. If no buffer is available when a packet arrives, the receiver enters a frame hunt state and discards each incoming packet from the current frame.
  - The UUI field is stored for the host into the RxBD after receiving the whole SSSAR frame.
- SSSAR transmitter
  - Segmentation of SSSAR SDUs from SSSAR TX queue into CPS packets
  - An SSSAR TX queue may contain several CIDs
  - Performs all the above mentioned CPS transmitter functions
  - A programmable segment length (Seg_Len) is copied from the SSSAR SDU into the CPS packet payload, except for when at the end of the frame or buffer.
  - UUI mode available. When UUI mode is enabled, the SSSAR UUI is copied from the byte following the last byte of the frame.

31.3 AAL2 Transmitter

The following sections describe the AAL2 transmitter.

31.3.1 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 31.3.2, “Transmit Priority Mechanism”) in order to fill the cell with new packets. The priority mechanism determines the order in which the TX queues are serviced. The transmitter continues to search for ready packets in the TX Queues until either the cell is successfully filled with packets, or no more packets are ready but the cell is not yet completed. In the first case the cell is simply sent. In the latter case, the optional Timer_CU (described in Section 31.3.5.1, “AAL2 Protocol-Specific TCT”) is examined. If the Timer_CU has expired, the uncompleted cell is padded with zeros and sent; otherwise, the cell is temporarily stored in external memory for the CP to attempt to complete it the next time the channel is scheduled.

The TX queues are the data structures that store the CPS packets and SSSAR frames. Each TX queue can contain different CIDs. Each TX queue is maintained by a Tx queue descriptor (TxQD) that holds the queue pointer and parameters to manage the queue.

When the transmitter fetches a packet out of an SSSAR TX queue, it usually takes out of the SSSAR buffer a number of octets equal to TxQD[Seg_Len] (see Section 31.3.5.4, “SSSAR Tx Queue Descriptor). The channel CID is taken from the BD of the first buffer of the SSSAR frame (see Section 31.3.5.5, “SSSAR Transmit Buffer Descriptor”). A CPS UUI = 27 is used for all the in-frame packets until the last packet from the SSSAR frame is sent. The last packet can optionally contain a per frame, user-defined UUI. After an SSSAR buffer is completely sent, an optional interrupt event is issued to the host. Also, if an SSSAR TX queue is empty an optional interrupt event is issued to the host.

In case of CPS TX queue, the transmitter fetches the packet header out of a buffer descriptor and the packet payload out of a CPS buffer (see Section 31.3.5.3, “CPS Buffer Structure”). The HEC in the packet header is calculated by the CP or taken from the buffer descriptor based on the user configuration. After a CPS packet is sent, an optional interrupt event is issued to the host. Also, if the CPS TX queue is empty, an optional interrupt event is issued to the host.

The optional partial filled mode (see Section 31.3.3, “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 31.3.4, “No STF Mode”) enables the transmission of cells that do not include the STF byte, thus allowing for 48-byte packets.

31.3.2 Transmit Priority Mechanism

The transmit priority mechanism operates in two modes:

- Round robin (TCT[Fix]=0)
- Fixed priority (TCT[Fix]=1)

The following sections describe the priority options.

31.3.2.1 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 31-4. 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.)

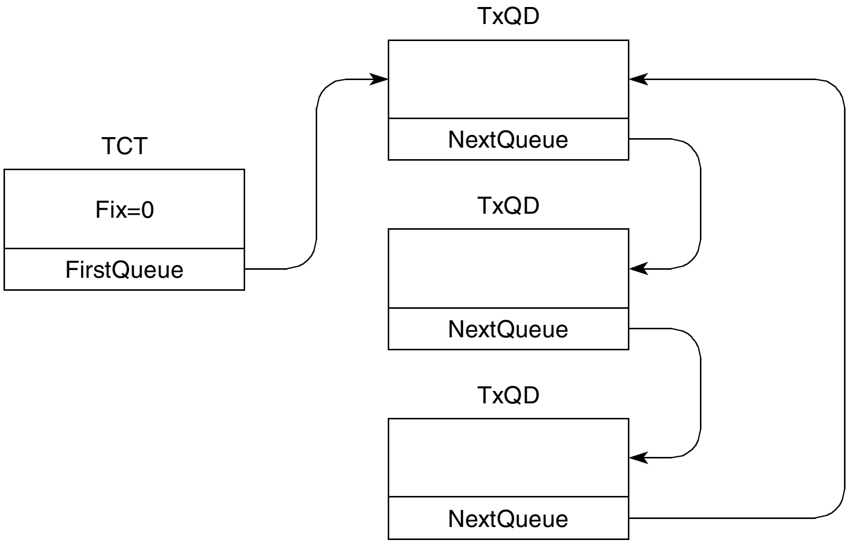


Figure 31-4. Round Robin Priority

The transmitter steps from one TxQD to the next along the queue links. The TCT[MaxStep] parameter limits the number of TX queues that the transmitter visits during a cell time. If MaxStep is reached before the cell has been completely filled, one of the following events takes place:

- TCT[ET]=0 (Timer CU disabled). The cell is padded with zeros and sent.
- TCT[ET]=1 (Timer CU enabled). If the timer has not expired, the cell is not sent. (The transmitter attempts to fill the cell the next time this channel is scheduled.) If the timer has expired, the cell is padded with zeros and sent.

After the transmitter sends a cell, it saves the queue link of the last TxQD serviced in TCT[FirstQueue].

31.3.2.2 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 31-5, and remains unchanged by the CP.

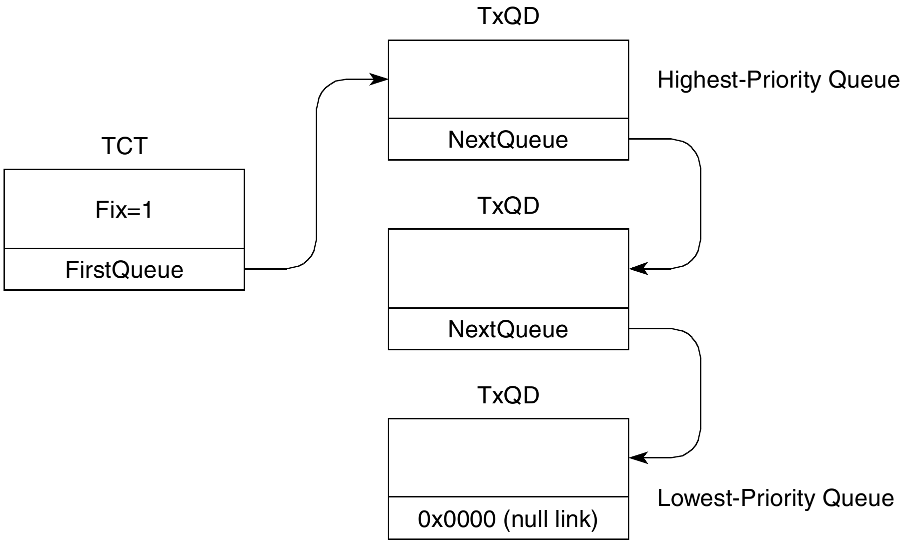


Figure 31-5. Fixed Priority Mode

The TCT[OneP] determines the number of packets that the transmitter attempts to take from each queue (see the explanation in round robin mode).

The NextQueue field of the lowest priority TxQD should be cleared (null link), and TCT[MaxStep] should be programmed to the total number of TX queues in the channel. When the transmitter reaches the null link and the cell is still not complete, the transmitter checks the TIMER CU mode as described above for the round robin mode.

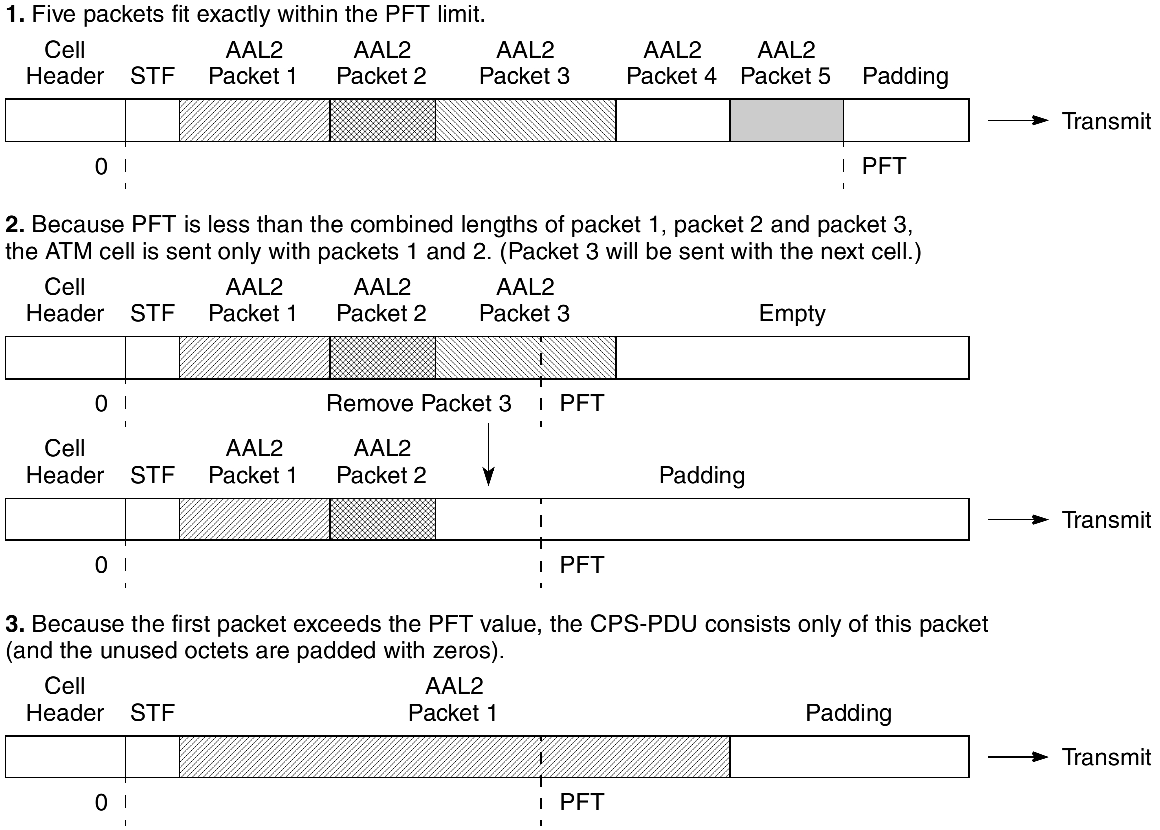
31.3.3 Partial Fill Mode (PFM)

The partial fill mode (TCT[PFM]=1) allows the user to specify a partial fill threshold (TCT[PFT]), which limits the number of data octets sent with each ATM cell. Partial fill mode assures that packets are not split over two cells, unless the first packet of the cell is greater than 47 bytes.

In partial fill mode, the transmitter starts by filling the ATM cell with the first packet. After the first packet, the CP determines if including the next packet in the cell would exceed the PFT limit. If so, the second packet is not inserted into the current cell, the unused payload is padded with zeros, and the cell is sent. If the second packet does not exceed PFT, the CP inserts it into the CPS-PDU and moves on to the third packet, and so on.

By programming PFT = 1, the partial fill mode can also serve to assure that only one packet is sent per cell.

The following figures provide examples of partial fill mode operation:



1. Five packets fit exactly within the PFT limit.

2. Because PFT is less than the combined lengths of packet 1, packet 2 and packet 3, the ATM cell is sent only with packets 1 and 2. (Packet 3 will be sent with the next cell.)

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).

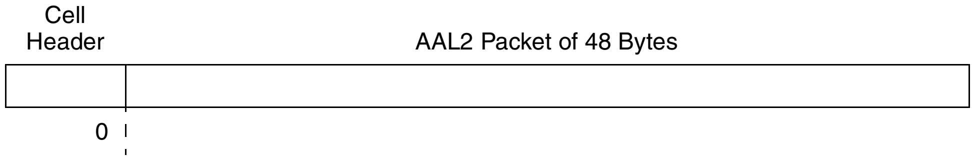
31.3.4 No STF Mode

The no-STF (no start of frame) mode enables the transmission of 48-byte packets by not including the STF byte in the CPS PDU. The no-STF mode should always be used with PFT programmed to 47 in order to prevent split and partial packets. For the AAL2 channels that use this mode, packets must not be larger than a maximum packet size of 48.

The user activates this mode per ATM channel by doing the following:

1. Setting TCT[NoSTF]=1 and TCT[PFM]=1
2. Establishing a partial fill threshold by programming TCT[PFT]=47

The following figure shows a cell using no-STF mode:



31.3.5 AAL2 Tx Data Structures

The following sections describe the transmit connection tables (TCT) and the structures in which CPS packets and SSSAR SDUs are stored in memory.

31.3.5.1 AAL2 Protocol-Specific TCT

The transmit connection table (TCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the TX queues of the ATM channel are maintained here. Figure 31-6 shows the AAL2-specific TCT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		—	DTB	BIB	AVCF	PFM	ATT		ET	VCON	INTQ	
Offset + 0x02	—												NoSTF	AAL		
Offset + 0x04	—															
Offset + 0x06																
Offset + 0x08	—															
Offset + 0x0A	FirstQueue															
Offset + 0x0C	Rate Remainder								PCR Fraction							
Offset + 0x0E	PCR															
Offset + 0x10	Timer_CU_period															
Offset + 0x12	Timer_Period_Shadow															
Offset + 0x14	—															
Offset + 0x16	APC Linked Channel															
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)															
Offset + 0x1a																
Offset + 0x1C	—		PMT						MaxStep							
Offset + 0x1E	—		PFT						—				OneP	STPT	Fix	PM

Figure 31-6. AAL2 Protocol-Specific Transmit Connection Table (TCT)

NOTE

When the channel is active, the CP fetches the TCT (32 bytes) using burst cycle and writes back only the first (24 bytes).

Table 31-1 describes the AAL2 TCT fields.

.

Table 31-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	**BO**	Byte ordering. This field is used for data buffers. 00 Reserved 01 PowerPC little endian 1x Big endian
	5	—	Reserved, should be cleared during initialization
	6	**DTB**	Data buffer bus selection. 0 Data buffers reside on the 60x bus. 1 Reserved
	7	**BIB**	Bus selection for the BDs, interrupt queues and the free buffer pool. 0 Reside on the 60x bus 1 Reserved.
	8	**AVCF**	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table, 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears the VCON bit. (Bit 13)
	9	**PFM**	Partial fill mode. See Section 31.3.3, "Partial Fill Mode (PFM)." 0 Partially filled cells are not supported. 1 Partially filled cells are supported.
	10-11	**ATT**	ATM traffic type 00 Peak cell-rate pacing (regular traffic). The host must initialize PCR and PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and Minimum cell rate pacing. The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved
	12	**ET**	Enable Timer_CU. 0 Timer_CU operation in this channel is disabled. 1 Timer_CU operation in this channel is enabled.
	13	**VCON**	Virtual channel is on. Should be set by the host before it issues an ATM TRANSMIT command. When the host sets the STPT (stop transmit) bit, the CP deactivates this channel and clears VCON. The host can issue another ATM TRANSMIT command only when the CP clears VCON.
	14–15	**INTQ**	Points to one of the four interrupt queues available

Table 31-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	0–11	—	Reserved, should be cleared during initialization
	12	**NoSTF**	No STF byte. See Section 31.3.4, “No STF Mode.” 0 Normal AAL2 cell structure 1 The cell does not include the STF byte. In this mode each cell starts with a new packet and contains only whole packets (no split or partial).
	13–15	**AAL**	AAL type 000 AAL0 —Segmentation with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol 010 AAL5 —ATM adaptation layer 5 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1_CES. Refer to Chapter 31, “ATM AAL1 Circuit Emulation Service.” All others reserved
0x04	—	—	Reserved, should be cleared during initialization
0x06	—	—	Reserved, should be cleared during initialization
0x08	—	—	Reserved, should be cleared during initialization
0x0A	—	**FirstQueue**	Points to the first queue to be serviced in the transmitter cycle. In round-robin priority mode (TCT[Fix]=0), this pointer could point to any one of the TxQDs related to this channel. In fixed priority mode (TCT[Fix]=1), this pointer should point to the highest priority TxQD.
0x0C	0–7	**Rate Remainder**	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared during initialization by the user.
	8–15	**PCR Fraction**	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	**PCR**	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	**Timer_CU_period**	Timer_CU duration in units of APC slots. Assures that CPS-Packet already packed in a cell wait at most the Timer_CU duration. The user defines this field.
0x12	—	**Timer_Period_Shadow**	This field must be initialized to the Timer_CU period in units of APC slots.
0x14	—	—	Reserved, should be cleared during initialization
0x16	—	**APCLC**	APC linked channel. Used by the CP. Should be cleared during initialization.
0x18	—	**ATMCH**	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.

Table 31-1. AAL2 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	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.
	8–15	**MaxStep**	Holds the number of TX Queues visited for each cell being prepared for transmission. In fixed priority mode (TCT[Fix]=1), MaxStep should be set to the total number of TX queues in the channel. See Section 31.3.2, "Transmit Priority Mechanism."
0x1E	0–1	—	Reserved, should be cleared during initialization.
	2–7	**PFT**	Partial fill threshold. Used for partially filled cells only; see Section 31.3.3, "Partial Fill Mode (PFM)." Specifies the maximum number of packet bytes allowed in a CPS PDU. The range 1–48 are valid values. If PFT=48 in partial fill mode, performance is adversely affected. When not in partial fill mode, PFT must be initialized to 47.
	8–11	—	Reserved, should be cleared during initialization
	12	**OneP**	One packet per queue. See Section 31.3.2, "Transmit Priority Mechanism." 0 The transmitter reads as many packets as possible from each TX queue before moving to the next queue. 1 The transmitter reads only one packet from each TX queue before advancing to the next queue.
	13	**STPT**	Stop transmit. Should be cleared during initialization. When the host sets this bit, the CP removes this channel from the APC and clears TCT[VCON] flag.
	14	**Fix**	Fixed priority. See Section 31.3.2, "Transmit Priority Mechanism." 0 Round robin priority. The TX Queues related to this channel all share the same priority. 1 Fixed priority. The TX Queues are ordered in a fixed priority ladder.
	15	**PM**	Performance monitoring 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

[1] **Boldfaced** entries must be initialized by the user.

31.3.5.2 CPS Tx Queue Descriptor

Each CPS TxBD table is managed by a CPS Tx queue descriptor (TxQD), as shown in Figure 31-7. 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 31.3.2, "Transmit Priority Mechanism." The CPS TxQD is located in the dual-port RAM, in a 16-byte aligned address.

	0 … 7	8	9	10	11	12	13 … 15
Offset + 0x00	—	**BNM**	**SW**	**HEC**	**CPS**	**TBM**	—
Offset + 0x02	**TxBD Table Offset In** (Switched Mode Only)						
Offset + 0x04	**TxBD Table Base**						
Offset + 0x06							
Offset + 0x08	**TxBD Table Offset Out**						
Offset + 0x0A	**Number of Packets In Queue**						
Offset + 0x0C	**NextQueue**						
Offset + 0x0E	—						

Figure 31-7. CPS Tx Queue Descriptor (TxQD)

Table 31-2 describes the CPS TxQD fields.

.

Table 31-2. CPS TxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	—	Reserved for internal use. (Used to save the BD status of the open BD.)
	8	**BNM**	Buffer not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**SW**	Switching queue. 0 Normal TX queue 1 This TxQD handles a switching queue. The receiver and transmitter share this queue.
	10	**HEC**	HEC calculation. 0 The transmitter calculates the CPS header HEC. 1 The CPS header HEC is taken as is from the CPS buffer descriptor.
	11	**CPS**	Sublayer type. For a CPS TxQD, this field must be set. 0 SSSAR or SSTED 1 CPS packet
	12	**TBM**	Transmit buffer interrupt mask. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue). 1 The transmit buffer event of this queue is enabled.
	13–15	—	Reserved, should be cleared during initialization
0x02	—	**TxBD Table Offset In**	Used only when this queue is used for switching (SW=1). Should be cleared during initialization.
0x04	—	**TxBD Table Base**	This pointer points to the base address of the BD table.
0x08	—	**TxBD Table Offset Out**	Holds the offset from the TxBD table base to the next BD to be opened by the transmitter. Should be cleared during initialization.

Table 31-2. CPS TxQD Field Descriptions (continued)

Offs et	Bits	Name[1]	Description
0x0A	—	**Number of Packets In Queue**	Counts the number of packets currently in the queue. If this queue is switched, the receiver increments this counter with each new received packet and the transmitter decrements it with each packet sent. For switching, the user should initialize this counter to zero. When this queue is not switched, this counter counts down with every packet sent. (This can have various purposes such as evaluating the packet rate that is transmitted from this queue.).
0x0C	—	**NextQueue**	Points to the next TxQD to be serviced after this one. See Section 31.3.2, "Transmit Priority Mechanism."
0x0E	—	—	Reserved, should be cleared during initialization

[1] **Boldfaced** entries must be initialized by the user.

31.3.5.3 CPS Buffer Structure

The CPS buffer structure consists of a BD table that points to data buffers. The BDs also contain, apart from the buffer pointer, the packet header. The buffers contain the packet payload. See Figure 31-8.

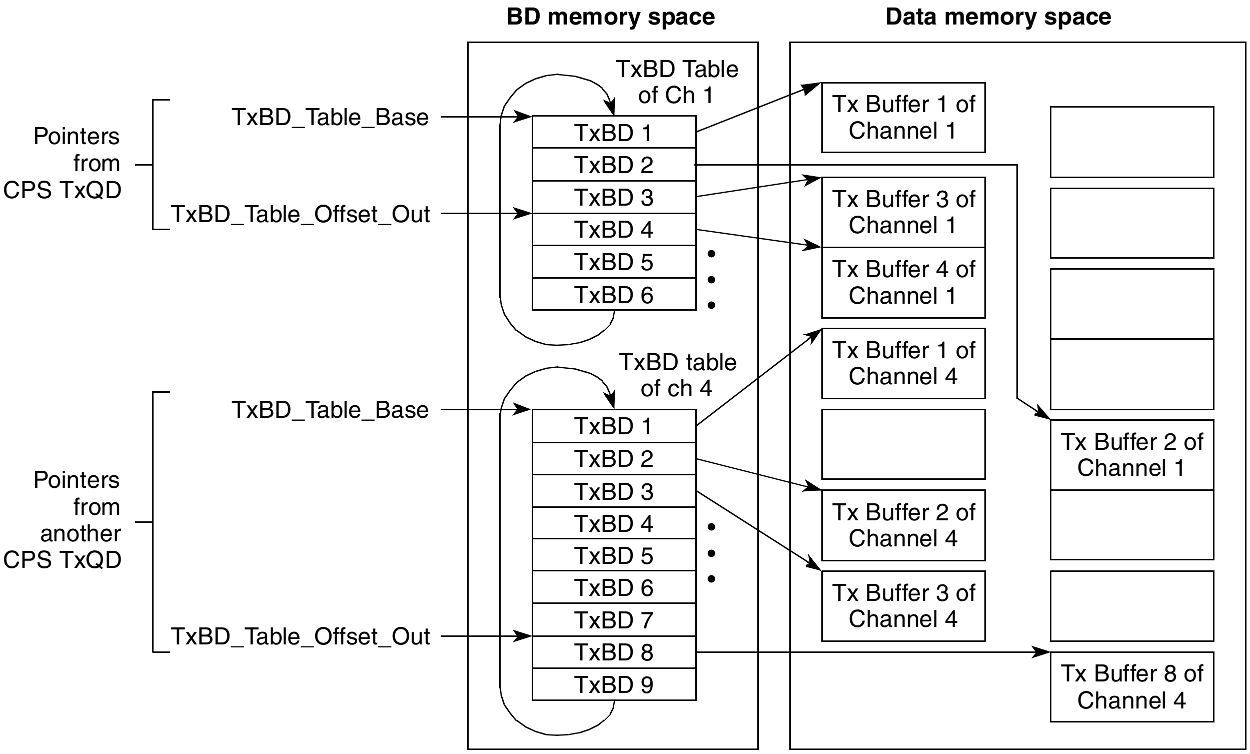


Figure 31-8. Buffer Structure Example for CPS Packets

Figure 31-9 shows a CPS TxBD.

	0	1	2	3	4–7	8–15
Offset + 0x00	**R**	**CM**	**W**	**I**	—	**CPS Packet Header**
Offset + 0x02	**CPS Packet Header**					
Offset + 0x04	**Tx Data Buffer Pointer (TXDBPTR)**					
Offset + 0x06						

Figure 31-9. CPS TxBD

Table 31-3 describes the CPS TxBD fields.

Table 31-3. CPS TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	**CM**[2]	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4–7	—	Reserved, should be cleared during initialization
	8–15	**CPS Packet Header**	This field contains the beginning (MSB) of the 3-byte packet header. See Figure 31-10 for the CPS packet header format.
0x02	—	**CPS Packet Header**	This field contains the rest of the packet header. If TxQD[HEC] = 0, the HEC part of the packet header is calculated by the CP, and the user may disregard the five least-significant bits of this field. See Figure 31-10 for the CPS packet header format.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This pointer is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

[2] Setting Continuous mode (TxBD[CM] = 1) is not allowed in CID switching mode.

0 … 7	8 … 13	14 … 18	19 … 23
Channel Identifier (CID)	Length Indicator (LI)	User-to-user ID (UUI)	Header Error Check (HEC)

Figure 31-10. CPS Packet Header Format

31.3.5.4 SSSAR Tx Queue Descriptor

A SSSAR TxBD table and its associated buffers are collectively called an SSSAR TX queue. Each SSSAR TX queue is managed by an SSSAR TxQD, as shown in Figure 31-11. 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 31.3.2, "Transmit Priority Mechanism." The SSSAR TxQD is located in the dual-port RAM in a 32-byte aligned address.

	0 … 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							
Offset + 0x0E	—							
Offset + 0x10								
Offset + 0x12								
Offset + 0x14								
Offset + 0x16								
Offset + 0x18								
Offset + 0x1a								
Offset + 0x1C								
Offset + 0x1E								

Figure 31-11. SSSAR Tx Queue Descriptor

Table 31-4 describes the SSSAR TxQD fields.

.

Table 31-4. SSSAR TxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	—	Reserved, should be cleared during initialization
	8	**BNM**	Buffer-not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	**UUI**	UUI insertion mode 0 UUI of last CPS packet is 0 1 UUI of last CPS packet is taken from the next byte after the end of the buffer[2]
	10	**INF**	Indicates the current state of the frame. 0 The next packet will be the first of a new frame. 1 Currently in the middle of the frame
	11	**CPS**	Sublayer type. For an SSSAR TxQD, this field must be cleared. 0 SSSAR or SSTED. 1 CPS packet.
	12	**TBM**	Transmit buffer interrupt mask for TxBD table. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue.) 1 The transmit buffer event of this queue is enabled.
	13	**SSSAR**	SSSAR bit 0 SSTED sublayer 1 SSSAR sublayer
	14–15	—	Reserved, should be cleared during initialization.
0x02	0–7	**Seg_Len**	Specifies the maximum length in bytes of the SSSAR PDU (excluding the packet header). Seg_Len is limited to 45 in NoSTF=1 mode. The CP always attempts to segment the SSSAR SDU according to this length, but not more than it.
	8–15	—	Reserved, should be cleared during initialization.
0x04	—	**TxBD Table Base**	Must be initialized to the first TxBD by the user.
0x08	—	**TxBD Table Offset Out**	Used to calculate the pointer to the next TxBD to be used for transmission. Should be cleared during initialization.
0x0A	—	—	Reserved, should be cleared during initialization.
0x0C	—	**NextQueue**	Points to the next TxQD to be serviced after this one. See Section Section 31.3.2, "Transmit Priority Mechanism."
0x0E–0x1E	—	—	Reserved, should be cleared during initialization.

[1] **Boldfaced** entries must be initialized by the user.

[2] The 5-bit UUI field is inserted into the header of the last packet of an SSSAR SDU. The user must append the UUI to the last buffer as an additional byte. This additional byte is then inserted into the UUI field of the last packet header (note that, it is important that this additional byte—the byte after the last byte in the last data buffer of the SSSAR frame—contains the 5 bits of the UUI). The 3 MSB bits of this extra byte should be cleared; refer to Figure 31-10.

31.3.5.5 SSSAR Transmit Buffer Descriptor

The SSSAR buffer structure consists of a BD table that points to data buffers. The buffers may contain SSSAR SDUs belonging to different CIDs. Each buffer may contain a whole SSSAR SDU or part of it. The CPS CID is located in the first BD of the SSSAR SDU. See Figure 31-12.

	0	1	2	3	4	5–7	8–15
Offset + 0x00	R	CM	W	I	L	—	CID
Offset + 0x02	Data Length (DL)						
Offset + 0x04	Tx Data Buffer Pointer (TXDBPTR)						
Offset + 0x06							

Figure 31-12. SSSAR TxBD

Table 31-5. SSSAR TxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**R**	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	**CM**	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	**L**	Last. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–7	—	Reserved, should be cleared during initialization.
	8–15	**CID**	Contains the CID number of the SSSAR SDU pointed by this BD. This field should be written to the first BD of an SSSAR SDU.

Table 31-5. SSSAR TxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	**Data Length**	Contains the length of the buffer associated with this BD. If this is the last buffer (L=1) and the UUI bit in the SSSAR TxQD is set, the 5-bit UUI field is located at (TXDBPTR + Data Length)[3–7] with bit [3] being the msb, that is, in the byte (right justified) immediately following the last byte of the buffer. For best bandwidth utilization and optimized partitioning of the SDU to packets of exactly Seglen size when an SDU is spread over multiple BD's, the application should set Data Length to be an integer multiple of Seg_Len. (Data Length == n x Seg_Len). For an SDU on a single BD this restriction does NOT apply.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

31.4 AAL2 Receiver

The following sections describe the AAL2 receiver.

31.4.1 Receiver Overview

The receiver cycle starts after the FCC receives a cell. If the cell header is successfully mapped to an ATM channel number, the corresponding RCT is fetched and the AAL type is read. For AAL2 cells, the receiver begins by checking if the last cell received in this channel (CID) has an uncompleted (split) packet. If so, the receiver first finishes handling this packet.

The receiver then goes through a validation process. The receiver matches the OSF field in the STF with the expected OSF based on the actual split packet (if the first packet is not split, the OSF should be zero). If the two values do not match, an OSF error interrupt is issued and the receiver drops the last packet. Also, if the STF parity check, the SN check or the OSF>47 check results in an error, the receiver issues an interrupt and discards the whole cell. If any of the above errors has occurred and the cell has started with the remainder of an uncompleted packet, the receiver does the following:

- For a CPS sublayer CID, the packet's RxBD[UP] (uncompleted packet) bit is set.
- For an SSSAR sublayer CID, the buffer is closed with RxBD[RxError = US = 10] (uncompleted SDU), and the rest of the frame is dropped.

The receiver now begins the process of extracting new CPS packets out of the cell with another round of error checking. The receiver examines each CPS packet header for the following errors:

- Incorrect packet HEC. The packet and rest of the cell are discarded.
- Packet length (LI + 1) is larger than CPS_Max_SDU_Length. The receiver discards the packet and 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 31.4.2, “Mapping of PHY | VP | VC | CID.” The match process yields an RxQD pointer. The RxQD indicates the type of SAR operation to be performed on the PHY | VP | VC | CID. Three SAR operation modes are supported:

- For the CPS sublayer, each packet from the CID is stored, as is, in a one packet buffer.
- For switching, the packet is stored directly into the transmit buffer, and the CID is translated.
- For the SSSAR sublayer, all packets received from the CID are reassembled into an SSSAR SDU similar to the BD and buffer structures used for AAL5 frames.
- For the SSSAR sublayer, last packet UUI indication is stored in the last RxBD of the SDU.

For the SSSAR sublayer, two additional parameters are verified for each new packet received:

- RAS_Timer expiration. The RAS_Timer_Duration defined in the AAL2 parameter RAM limits the time (starting when the first packet is received) allowed to receive a complete SSSAR SDU. If this time limit is exceeded, the receiver closes the current buffer with RxBD[RxError = TE = 01] (Ras_Timer expired) and starts a new SSSAR frame with the next packet. When a buffer is closed with RxBD[RxError = TE = 01], RxBD[L] is not set and the receiver does not issue an RXF interrupt.
- SSSAR_Max_SDU_Length. With each new packet the receiver checks whether the current accumulated length of the SSSAR SDU exceeds the SSSAR_Max_SDU_Length. If so, the receiver closes the current buffer with RxBD[RxError = OS = 11] (oversized), discards the rest of the SSSAR SDU, RxBD[L] is set, and if RxQD[RFM]=1 issues an RXF interrupt.

NOTE

CIDs that have the same number but that are from different AAL2 connections cannot use the same RxQD, unless they never have split packets.

31.4.2 Mapping of PHY | VP | VC | CID

The AAL2 mapping mechanism translates a PHY | VP | VC | CID combination into an RxQD. An RxQD can be unique per PHY | VP | VC | CID.

The mapping mechanism, shown in Figure 31-13, can be broken down as follows:

- Each ATM channel number (RCT) has its own CID mapping table. The mapping table can be placed in internal or external memory (according to RCT[MAP]) and is pointed to by RCT[CID Mapping Table Base]. The CID of the received packet is used as an index into the mapping table.
- Each entry in the mapping table contains a 2-byte RxQD offset. This offset, multiplied by 4, is the offset to an RxQD in either the internal or external RxQD table.
- The two RxQD tables serve all the ATM channel numbers of an FCC. (RxQD_Base_Int and RxQD_Base_Ext are defined in the FCC parameter RAM.)

- RxQD offsets from 8 through 511 point into the internal RxQD table located in dual-port RAM at RxQD_Base_Int. Note that the first 32 bytes of the internal RxQD table are reserved (so offsets 0–7 are reserved).
- RxQD offsets greater than 511 point into the external RxQD table located at RxQD_Base_Ext + (512 × 4).
- Because the three types of RxQDs are different sizes, some offset numbers may not be used.

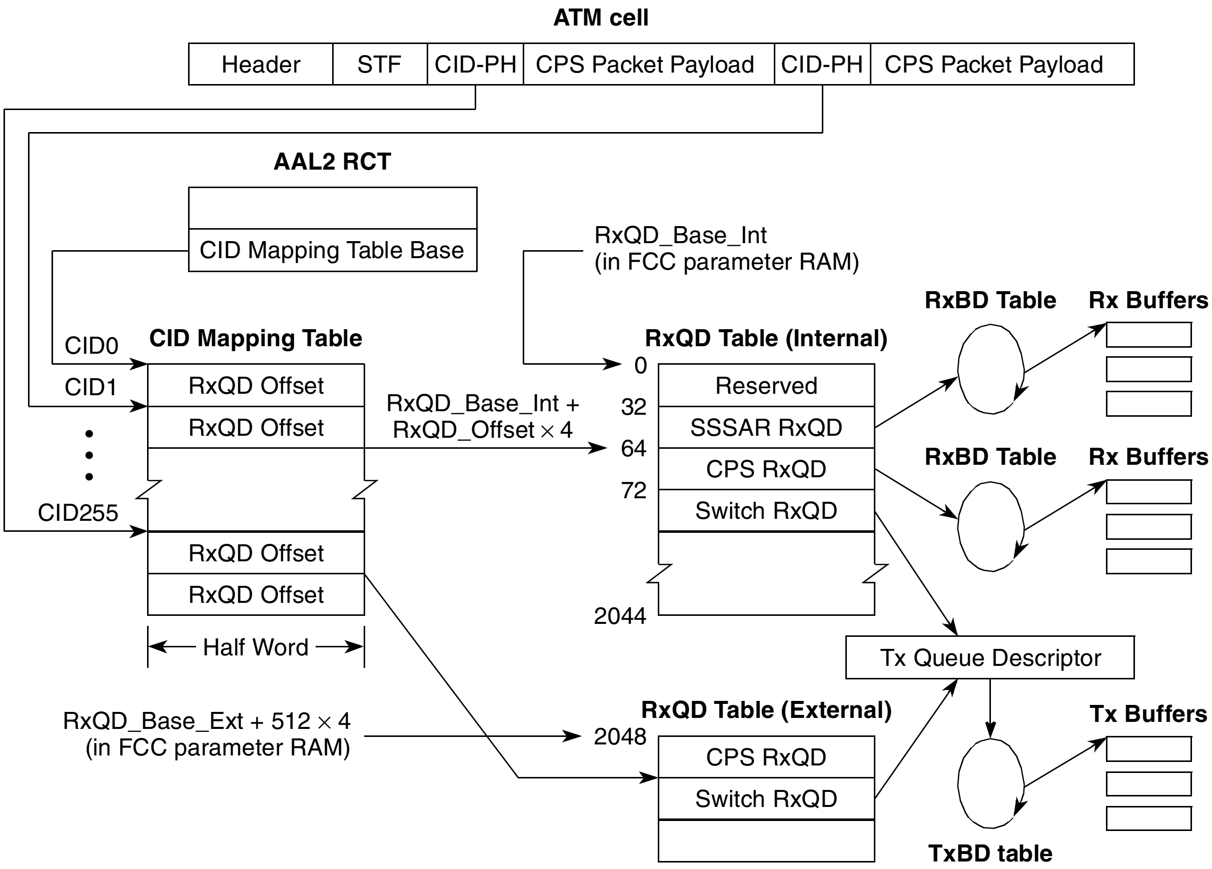


Figure 31-13. CID Mapping Process

31.4.3 AAL2 Switching

Switching is performed by pointing an RX CID at a switch RxQD (see Figure 31-14). The switch RxQD is unique for each Rx CID. The descriptor holds a translation CID number and a pointer to a CPS TxQD into which this packet is saved and later sent by the transmitter. (The TxQD pointer is responsible for the actual PHY | VP | VC switching.) The TxQD pointed to by the switch RxQD(s) should have TxQD[SW] set and should not be modified by the host when the channel is active. The transmit scheduling of the packet is done by the APC according to the programmed bit rate of the ATM channel that holds the switched queue.

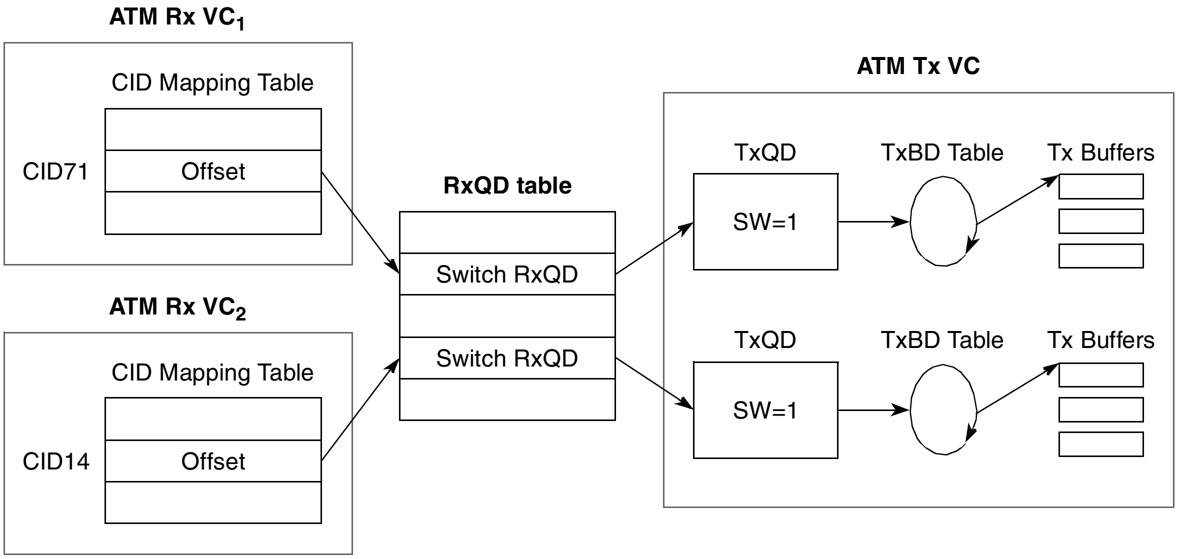


Figure 31-14. AAL2 Switching

A partial packet discard mode is provided for the AAL2 switched channels that perform end-to-end SSSAR. When this mode is enabled (switch RxQD[PPD]=1), if no buffer is available to receive a packet in the middle of a frame, the subsequent middle packets of the SSSAR SDU are discarded. When the last packet of the SSSAR SDU arrives, the receiver attempts to reopen a buffer.

A number-of-packets-in-queue counter is available in the TxQD. The CP increments the counter for each packet received and decrements it for each packet sent. The host can poll the counter periodically to verify that the switching queues are not overloaded.

On any open BD that is partially filled, the receiver sets the UP (uncomplete packet) bit. When the packet is fully received during a normal error-free operation, the UP mark is removed, the Empty bit is set, and operation continues. However, if an error is detected by the receiver, the Empty bit is set and the UP bit remains set. In such a case, the transmitter skips this BD and proceeds to the next one. If for any reason the receiver that was in the middle of the BD stopped receiving traffic, the UP bit remains set and the Empty bit is cleared. Another receiver using the same BD ring monitors the UP bit in addition to the Empty bit. If the UP bit is set, the other receiver does not proceed with the reception and gives a Busy interrupt. See Figure 31-17.

The receiver that is in a 'stuck' state marks the BD with the receiver channel code that received the partial packet so that the host intervention is easier if needed. See Figure 31-19.

If the TBNR Time Out CNT mechanism is used, the transmitter advances after it tries to transmit the same BD with UP set after a given amount of attempts. The BD will be freed up for use. Refer to the TBNR Time Out CNT description in Table 31-6.

31.4.4 AAL2 RX Data Structures

The following sections describe the receive connection tables and the structures in which CPS packets and SSSAR SDUs are stored in memory.

31.4.4.1 AAL2 Protocol-Specific RCT

The receive connection table (RCT) is the VC-level table 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 31-15 shows the AAL2-specific RCT.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		—	DTB	BIB	—		SEGF	ENDF	—	MAP	INTQ	
Offset + 0x02	—												NoSTF	AAL		
Offset + 0x04	—															
Offset + 0x06																
Offset + 0x08																
Offset + 0x0A																
Offset + 0x0c																
Offset + 0x0e																
Offset + 0x10																
Offset + 0x12																
Offset + 0x14																
Offset + 0x16																
Offset + 0x18	CID Mapping Table Base															
Offset + 0x1A																
Offset + 0x1C	—		PMT						TBNR Time OUT CNT							
Offset + 0x1E	Max_CPS_SDU_Deliver_length								—						EM	PM

Figure 31-15. AAL2 Protocol-Specific Receive Connection Table (RCT)

NOTE

> For an active channel, the CP uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes.

Table 31-6 describes the AAL2 RCT fields.

Table 31-6. AAL2 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	Description
0x00	0–1	—	Reserved, should be cleared during initialization
	2	**GBL**	Global. Setting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	**BO**	Byte ordering—used for data buffers. 00 Reserved 01 Munged little endian 1x Big endian
	5	—	Reserved, should be cleared during initialization
	6	**DTB**	Data buffer bus selection. 0 Reside on the 60x bus 1 Reserved.
	7	**BIB**	Bus selection for the BDs, interrupt queues, CID mapping table, RxQDs, and the free buffer pool. 0 Reside on the 60x bus 1 Reserved
	8-9	—	Reserved, should be cleared during initialization
	10	**SEGF**	OAM F5 segment filtering 0 Do not send cells with PTI=100 to the raw cell queue. 1 Send cells with PTI=100 to the raw cell queue.
	11	**ENDF**	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12	—	Reserved, should be cleared during initialization
	13	**MAP**	CID mapping table memory location select 0 Resides in the dual-port RAM 1 Resides in external memory
	14–15	**INTQ**	Assigns one of the four available interrupt queues to this ATM channel number
0x02	0–11	—	Reserved, should be cleared during initialization
	12	**NoSTF**	No STF byte. 0 Normal AAL2 cell structure 1 The cell does not include the STF byte. In this mode each cell starts with a packet and contains only whole packets (no split or part).
	13–15	**AAL**	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol 010 AAL5 —ATM adaptation layer 5 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1_CES. Refer to Chapter 31, “ATM AAL1 Circuit Emulation Service.” All others reserved

Table 31-6. AAL2 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	Description
0x04	—	—	Reserved, should be cleared during initialization
0x06	—		
0x08	—		
0x0A	—		
0x0C	—		
0x0E	—		
0x10	—		
0x12	—		
0x14	—		
0x16	—		
0x18	—	**CID Mapping Table Base**	Points to the base address of the CID mapping table (see Figure 31-13). If RCT[MAP] = 0, the pointer contains a dual-port RAM address and only the 16 lsb (at 0x1A and 0x1B) are relevant. If MAP = 1, the pointer is a full 32-bit address in the memory space.
0x1C	0–1	—	Reserved, should be cleared during initialization
	2–7	**PMT**	Performance monitoring table. Assigns one of the available 64 performance monitoring tables to this VC. The table's starting address is PMT_BASE+PMT × 32.
	8–15	**TBNR Time Out CNT**	The TBNR Time-Out CNT is a parameter that describes the amount of attempts the transmitter tries to transmit a packet on a BD ring which is current marked as partially filled, that is, waiting for a receiver to finish reception of a packet. This value will be used internally by the transmitter that is the destination for this packet and decremented by the CPM on each attempt. Upon reaching the value 1, the transmitter will act as if the receiver is stuck in error condition and proceed to the next BD in the BD ring. This parameter is valid in switch mode only and should be programmed to a higher value than the ratio between the transmitter rate and the lowest receiver rate in the BD ring. The 8 bit value is scaled by 4 (setting TBNR Time-Out CNT =1 yields a value of 4 internally) so that the max number is 1K. Clearing this field will disable this feature completely
0x1E	0–7	**Max_CPS_SDU_Deliver_Length**	Indicates the maximum size CPS_SDU in bytes that is allowed to be transported on this channel. This value is compared to the length of each CPS_SDU before it is delivered, as specified in the ITU-T recommendation I.363.2.
	8–13	—	Reserved, should be cleared during initialization
	14	**EM**	Receive error mask for AAL2 protocol-specific events. Note that buffer-not-ready, Rx buffer and Rx frame events are not affected by this mask. 0 Disable AAL2 receive error events. 1 Enable AAL2 error events.
	15	**PM**	Enable performance monitoring 0 No performance monitoring for this VC 1 Perform performance monitoring for this VC. Whenever a cell is received by this VC, the associated performance monitoring table is updated.

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.2 CID Mapping Tables and RxQDs

Each PHY | VP | VC | CID combination is assigned an RxQD using a CID mapping table. To multiplex several receive CIDs into a single common queue, map each multiplexed PHY | VP | VC | CID combination to one RxQD.

The ATM channel’s RCT contains the base address of the associated CID mapping table. This base address is external (32 bits) when RCT[MAP]=1; otherwise, the table resides in the dual-port RAM and the base address is two bytes. The CID of the received packet is used as an index into the mapping table. The mapping table entries are 2-byte RxQD offsets. If the CID mapping table is external, it must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

There are two RxQDs—one for internal RxQDs and one for external RxQDs. Offsets between 0–511 belong to the 2048-byte internal RxQD table. It is recommended to have as many RxQDs as possible in the internal table. Note that the first 32 bytes of the internal RxQD table are reserved for internal use; that is, RxQD offsets between 0–7 are reserved. The address of an internal RxQD is RxQD_Base_Int + 4 × RxQD_Offset.

Offsets between 512–65535 belong to the external RxQD table. The address of an external RxQD is RxQD_Base_Ext + 4 × RxQD_Offset. The external RxQD table must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

Because the three kinds of RxQDs are each a different size (for example, an SSSAR RxQD is 32 bytes and a CPS switch RxQD is only 4 bytes), some of the offset numbers are left unused.

31.4.4.3 CPS Rx Queue Descriptors

Each CPS RxQD, as shown in Figure 31-16, points to an CPS RxBD table.

	0 11	12	13	14 15
Offset + 0x00	—	RBM	—	SubType
Offset + 0x02	RxBD Table Offset			
Offset + 0x04	RxBD Table Base			
Offset + 0x06				

Figure 31-16. CPS Rx Queue Descriptor

Table 31-7 describes the CPS RxQD fields.

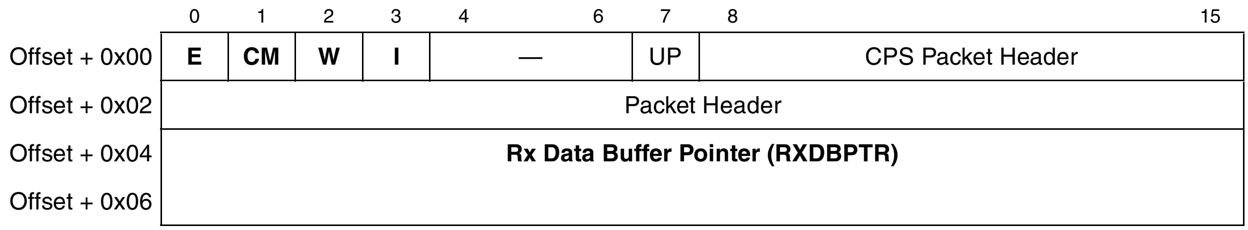
Table 31-7. CPS RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–11	—	Reserved, should be cleared during initialization
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	13	—	Reserved, should be cleared during initialization
	14-15	**SubType**	SubType. Sublayer type, should be 00 (CPS) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved
0x02	—	**RxBD Table Offset**	Holds the offset to the next BD to be opened by the receiver. Should be cleared during initialization.
0x04	—	**RxBD Table Base**	Holds the pointer to the first BD in the BD table

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.4 CPS Receive Buffer Descriptor (RxBD)

The CPS RxBD structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 31-17. The buffers contain the packet payload.



	0	1	2	3	4 – 6	7	8 – 15
Offset + 0x00	**E**	**CM**	**W**	**I**	—	UP	CPS Packet Header
Offset + 0x02	Packet Header						
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**						
Offset + 0x06							

Figure 31-17. CPS Receive Buffer Descriptor

Table 31-8 describes the CPS RxBD fields.

Table 31-8. CPS RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Buffer empty bit 0 The CPS RX buffer is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E remains zero. 1 The CPS RX buffer is empty or reception is in progress. This is controlled by the CP. Once E is set, the core should not access any fields of this buffer.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 The CP will not issue an interrupt after this buffer is serviced. 1 The CP will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4–6	—	Reserved, should be cleared during initialization
	7	UP	Uncompleted packet 0 No error occurred in this packet 1 A receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 31-10 for the CPS packet header format.
0x02	—	CPS Packet Header	Contains the rest of the packet header. The CP checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 31-10 for the CPS packet header format.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.5 CPS Switch Rx Queue Descriptor

The switch RxQD, shown in Figure 31-18, is used for CIDs that are being switched from one PHY_1 | VP_1 | VC_1 | CID_1 to another PHY_2 | VP_2 | VC_2 | CID_2. The RxQD contains the pointer to the TxQD that controls the TxBD table through which the packet is transferred.

The switch RxQD also contains the translation CID that is saved with the packet in the transmit buffer. A PPD mode enables the discarding of the rest of an SSSAR frame when a buffer is not available.

Note that the CPS switch RxQD must be unique for every Rx switched CID.

	0 — 7	8 — 11	12	13	14 — 15
Offset + 0x00	**TX CID**	—	**RBM**	**PPD**	**SubType**
Offset + 0x02	**TxQD Pointer**				

Figure 31-18. CPS Switch Rx Queue Descriptor

Table 31-9 describes the CPS switch RxQD fields.

Table 31-9. CPS Switch RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–7	**TX CID**	Translation CID. The received CID is saved in a TX queue with this new CID number.
	8–11	—	Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	**PPD**	Partial packet discard 0 Normal mode 1 When a buffer-not-ready event causes a packet to be discarded, the remainder of the SSSAR SDU is also discarded. This allows for better performance for switched channels that implement SSSAR.
	14–15	**SubType**	Sublayer type. Should be 01 (CPS switched) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved
0x02	—	**TxQD Pointer**	Points to the TxQD into which the packets of this CID are stored and later sent.

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.6 SWITCH Receive/Transmit Buffer Descriptor (RxBD)

The switch buffer structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 31-19. The buffers contain the packet payload. This BD is common to the receiver and the transmitter.

	0	1	2	3	4 — 6	7	8 — 15
Offset + 0x00	**E/R**	**0**	**W**	**I**	—	UP	CPS Packet Header
Offset + 0x02	Packet Header (Receiver CC)						
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**						
Offset + 0x06							

Figure 31-19. Switch Receive/Transmit Buffer Descriptor

Table 3-11 describes the switch RxBD fields.

Table 31-10. Switch RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E/R**	Buffer ready Must be set to zero
	1	**0**	Not valid for switching mode, should be cleared to 0 upon initialization.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt. 0 The CP will not issue an interrupt after this buffer is serviced. 1 The CP will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4–6	—	Reserved, should be cleared during initialization
	7	UP	Uncompleted packet. 0 No error occurred in this packet and the complete packet has been received. 1 if R/E=1 a receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue. The transmitter will skip this BD when in this state and continue to the next BD in the ring. If R/E=0 a receiver has received the first part of a packet and is waiting for the rest of it to be received on the next ATM cell.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 31-10 for the CPS packet header format. (see remark in next row)
0x02	—	CPS Packet Header (Receiver CC)	Contains the rest of the packet header. The CP checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 31-10 for the CPS packet header format. In case of a “stuck” receiver in switch mode, where the BD ring in common to Tx and Rx, this field indicates the last Receiver Channel Code number which has been received. The terminology for ‘stuck’ implies a receiver which started receiving a packet and the rest of the packet hasn’t been received.When the receiver is in a ‘stuck’ state the entry: CPS Packet Header is not valid. If the Time-out mechanism is being used this field is being used internally by the CPM.
0x04	—	**RXBDPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.7 SSSAR Rx Queue Descriptor

The SSSAR RxQD, as shown in Figure 31-20, 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 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 31-20. SSSAR Rx Queue Descriptor

Table 31-11 describes the SSSAR RxQD fields.

Table 31-11. SSSAR RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–10	—	Reserved, should be cleared during initialization
	11	**RasT**	Ras timer enable. 0 Ras timer disabled (timestamp 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 Ras timer 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 31-11. SSSAR RxQD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x02	—	**RxBD Table Offset**	Points to the next BD to be handled by the CP. The user should initialize this pointer to zero.
0x04	—	**RxBD Table Base**	Points to the beginning of the BD table
0x08	—	—	Reserved, should be cleared during initialization
0x0A	—	—	Reserved, should be cleared during initialization
0x0C	—	Time Stamp	Used for reassembly timeout of the SSSAR SDU. Whenever the first packet of an SSSAR SDU arrives the timestamp timer is sampled and stored here (regardless of the RasT bit).
0x10	—	—	Reserved, should be cleared during initialization
0x12	—	—	Reserved, should be cleared during initialization
0x14	—	**MRBLR**	Maximum receive buffer length. Holds the maximum receive buffer length. The actual buffer size can be less.
0x16	—	**Max_SSSAR_SDU_ Length**	Holds the maximum SSSAR SDU length. Upon each new packet the accumulated frame size is compared with this value. If the limit is exceeded, the CP discards the rest of the packets of the current frame.
0x18–0x1E	—	—	Reserved, should be cleared during initialization

[1] **Boldfaced** entries must be initialized by the user.

31.4.4.8 SSSAR Receive Buffer Descriptor

The SSSAR SDU is stored in a BD-buffer structure similar to the structures used for AAL5 frames. The buffer size is determined by SSSAR RxQD[MRBLR]; the actual buffer space used may be smaller. If the received SSSAR SDU is greater than MRBLR, the SDU spans over multiple buffers.

The SSSAR RxBD is shown in Figure 31-21.

	0	1	2	3	4	5	6	7	10	11	15
Offset + 0x00	**E**	**CM**	**W**	**I**	L	**RxError**		—		UUI	
Offset + 0x02	Data Length (DL)										
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**										
Offset + 0x06											

Figure 31-21. SSSAR Receive Buffer Descriptor

Table 31-12 describes the SSSAR RxBD fields.

Table 31-12. SSSAR RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E**	Empty 0 The buffer associated with this RxBD is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in the table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 An Rx buffer event is sent (provided that RxQD[RBM] is set) to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	L	Last. Set by the CP. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–6	RxError	Rx error occurred 00 No Rx error occurred 01 TE—Ras_Timer expired. The Ras timer expired before this buffer could be completed. The SSSAR SDU stored in this buffer is not completed.[2] 10 US—Uncompleted SDU. A receive error caused a packet belonging to this SSSAR SDU to be lost. The receiver discarded the rest of this SSSAR SDU. 11 OS—Oversized. The size of the SSSAR SDU has exceeded the Max_SSSAR_SDU_Size parameter. The rest of the SDU was discarded.
	7–10	—	Reserved, should be cleared during initialization
	11–15	UUI	Contains the UUI of the last packet in the received SDU. Valid only where the L bit is set.
0x02	—	Data Length	Contains the length of the buffer associated with this BD. If this is the last buffer (L=1) of the SSSAR SDU, this field contains the total frame length.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] **Boldfaced** entries must be initialized by the user.

[2] When RAS timer expires the RxBD is closed with RAS timer expired indication, the Last (L) bit is not set, and RXF interrupt is not issued. A new RxBD is opened for the next incoming AAL2 packet and the frame is processed as normal and is treated as a new frame. When the next SSSAR end-of-frame indication is received, the RxBD at that time is closed with an L indication, and if RxQD[RFM] = 1, the receiver issues an RXF interrupt.

31.5 AAL2 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 31-13. The table includes both the fields for general ATM operation and also the fields specific to AAL2 operation.

Note that some of the values must be initialized by the user, but values updated by the CP should not be modified by the user.

Table 31-13. AAL2 Parameter RAM

Offset	Name	Width	Description
0x00–0x3F	—	—	Reserved. Should be cleared during initialization.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User-defined.
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined.
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 32 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined.
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined.
0x4C	RAS_Timer_Duration	Word	Contains the RAS_Timer duration in microseconds for the SSSAR sublayer. User-defined.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. The offset of the UEAD entry in the UDC extra header. Should be an even address. If RCT[BO]=01 UEAD_OFFSET should be in little-endian format. For example if UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be set to 2 (second half word entry in internal RAM).
0x5E	RxQD_Base_Int	Hword	Points to the base address of the internal RxQD table. The pointer should be 32-byte aligned. User-defined.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined.
0x62	APCP_BASE	Hword	APC parameters table base address. User-defined.
0x64	FBT_BASE	Hword	Free buffer pool parameters table base. User-defined.

Table 31-13. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0x66	INTT_BASE	Hword	Interrupt queue parameters table base. User-defined.
0x68	—	—	Reserved. Should be cleared during initialization.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0-7] hold the 8 most significant bits of the RX/TxBD table base address. BD_BASE_EXT[8-31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCI_Filtering	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7-15 are received and the associated VCI_Filtering bit = 1 the cell is sent to the raw cell queue. VCI=3 is associated with VCI_Filtering[3], VCI=15 is associated with VCI_Filtering[15]. VCI_Filtering[0–2, 5] should be zero. See Section 30.10.1.2, “VCI Filtering (VCIF).”
0x84	GMODE	Hword	Global mode. User-defined. See Section 30.10.1.3, “Global Mode Entry (GMODE).”
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 30.15, “ATM Transmit Command. “
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved. Should be cleared during initialization.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFFFFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SN protection look up table base address. (AAL1 only.) The 32-byte table resides in dual-port RAM and must be initialized by the user (See Section 30.10.6, “AAL1 Sequence Number (SN) Protection Table”).
0x9A	—	Hword	Reserved. Should be cleared during initialization.
0x9C	SRTS_BASE	Word	External SRTS logic base address. (AAL1 only.) Should be 16-byte aligned.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/Unassign cell size. 52 in regular mode. 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A6A6A.

Table 31-13. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler (See RTSCR). For time stamp prescaler of 1 μs, Trm should be set to 100 ms/1 μs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined.
0xB4	RxQD_Base_Ext	Word	Points to the base address of the external RxQD table. The actual address of the first RxQD in the table is RxQD_Base_Ext + 512 × 4. User-defined.
0xB8	RX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the RX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: RX_UDC_Base + 16 × CH# (where CH# is the ATM channel number)
0xBC	TX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the TX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: TX_UDC_Base + 16 × CH# (where CH# is the ATM channel number)
0xC0–0xDF	—	—	Reserved. Should be cleared during initialization.
0xE0	TCELL_TMP_BASE_EXT	Word	Transmit cell temporary base address. Points to a total of 64 × last_AAL2_Ch# octets reserved in external memory for partially filled cells. Note: TCELL_TMP_BASE_EXT must be on the same bus as the all the AAL2 data buffers required for CPS, SSSAR and CID switching.
0xE4–0xFB	—	—	Reserved. Should be cleared during initialization.
0xFC	PAD_TMP_BASE	Hword	PAD template base address. Points to an internal memory area that contains the zero cell template. Should be 64-byte aligned. User-defined.
0xFE	—	—	Reserved. Should be cleared during initialization.

31.6 User-Defined Cells in AAL2

The user-defined cell (UDC) mode for ATM as described in Section 30.7, “User-Defined Cells (UDC),” also applies to AAL2 operation. However, for AAL2 operation only, the UDC headers reside in a table in external memory, not in the BDs.

For transmit channels in AAL2 UDC mode, initialize its UDC header entry in the TX UDC header table before activating the channel. The header can be up to 12 bytes. The TX_UDC_Base parameter in the parameter RAM (see Table 31-13), points to the beginning of the TX UDC header table.

The UDC header of a specific AAL2 transmit VC is located at the following address:

TX_UDC_Base + CH# × 16 (where CH# is the ATM channel number)

For receive channels in AAL2 UDC mode, the receiver copies the UDC header from the first cell received by the VC to the RX_UDC header table. The UDC headers of subsequent cells of that VC are discarded; UDC extended address mode (UEAD) is not affected.

The UDC header of a specific AAL2 receive VC is located at the following address:

RX_UDC_Base + CH# × 16 (where CH# is the ATM channel number)

The structure of a UDC header table (receive or transmit) is shown in Figure 31-22.

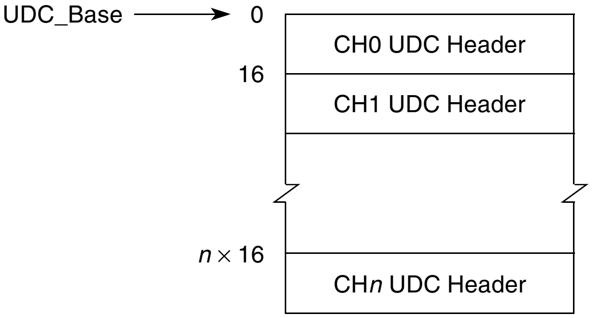


Figure 31-22. UDC Header Table

31.7 AAL2 Exceptions

For each VC, four circular interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ] for each VC, the user assigns an interrupt queue number.

When one of the CIDs generates an interrupt request, the CP writes a new entry to the interrupt queue containing the ATM channel number, the CID and a description of the exception. Because CID = 0 is a unique CID number, it is used to specify that the event is related to the VC rather than the CID. As with all ATM exceptions, the valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches a location with the W bit set, it wraps to the first entry in the queue. More details can be found in Section 30.11, “ATM Exceptions.”

An interrupt entry for a CID is shown in Figure 31-23.

	0	1	2	3–10	11	12	13	14	15
Offset + 0x00	**V**	—	**W**	CID	TBNR	RXB	BSY	TXB	RXF
Offset + 0x02	Channel Code (CC)								

Figure 31-23. AAL2 Interrupt Queue Entry CID ≠ 0

Table 31-14 describes the interrupt queue entry fields for a CID.

Table 31-14. AAL2 Interrupt Queue Entry CID ≠ 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. The exception occurred for this CID.
	11	TBNR	Tx buffer not ready interrupt. This interrupt is issued when the CP tries to open a TxBD, which is not ready (R = 0). This interrupt is sent only if TxQD[BNM] = 1. The interrupt has an associated channel code and CID. **Note:** The CID number that is placed in the interrupt queue is the one currently located in the last BD. Because the CID is not updated when the BD is not ready, the CID value is the one extracted from this BD when it was last processed and transmitted. If the BD is never processed and the BD was cleared, the CID value could be zero.
	12	RXB[1]	Rx buffer interrupt. This interrupt is issued when the I bit is set for an RxBD and the RxQD[RBM] bit is set. This interrupt has an associated channel code and CID.
	13	BSY	Busy condition. The RxBD table associated with this channel's CID is busy. Packets were discarded due to this condition.
	14	TXB	Transmit buffer interrupt. This interrupt is issued when the TxBD[I] bit is set. This interrupt is sent only if TxQD[TBM] is set. This interrupt has an associated channel code and CID.
	15	RXF[1]	Receive SSSAR SDU (frame). An SSSAR frame belonging to this channel's CID has been received. This interrupt is sent only if RxQD[RFM]=1.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

[1] These interrupt queue fields are defined differently for other AAL types. Refer toTable 30-41 for more information.

An interrupt entry for the VC is shown in Figure 31-24.

	0	1	2	3–10	11	12–15
Offset + 0x00	**V**	—	**W**	0000_0000		Error_Code
Offset + 0x02	Channel Code (CC)					

Figure 31-24. AAL2 Interrupt Queue Entry CID = 0

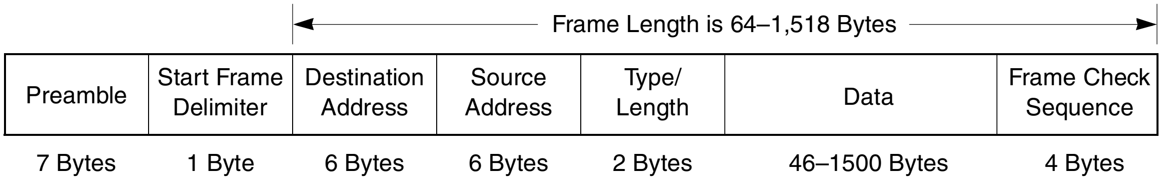
Table 31-15 describes the interrupt queue entry fields for the VC. All the receive error events are enabled by setting RCT[EM].

Table 31-15. AAL2 Interrupt Queue Entry CID = 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. Equals zero. This exception applies to the whole cell.
	11	—	Reserved
	12–15	Error_Code	A receive error was detected. 0000 Parity error of the OSF 0001 The STF sequence number is incorrect. 0010 The number of octets expected to overlap into this cell does not match the OSF. 0011 OSF is greater than 47. 0100 A packet HEC error was detected. 0101 The length of the CPS packet exceeds the Max_SDU_Length. 0111 A packet HEC error was detected in a split header packet.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

Chapter 32
Fast Ethernet Controller

The Ethernet IEEE 802.3 protocol is a widely-used LAN based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 32-1.



Frame Length is 64–1,518 Bytes

Preamble	Start Frame Delimiter	Destination Address	Source Address	Type/ Length	Data	Frame Check Sequence
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46–1500 Bytes	4 Bytes

Note: The lsb of each octet is transmitted first.

Figure 32-1. Ethernet Frame Structure

The elements of an Ethernet frame are as follows:

- 7-byte preamble of alternating ones and zeros
- Start frame delimiter (SFD)—Signifies the beginning of the frame
- 48-bit destination address
- 48-bit source address. Original versions of the IEEE 802.3 specification allowed 16-bit addressing, which has never been used widely.
- Ethernet type field/IEEE 802.3 length field. The type field signifies the protocol used in the rest of the frame, such as TCP/IP; the length field specifies the length of the data portion of the frame. For Ethernet and IEEE 802.3 frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This requirement limits the length of the data portion of the frame to 1,500 bytes and, therefore, the total frame length to 1,518 bytes.
- Data
- Four-bytes frame-check sequence (FCS), the standard, 32-bit CCITT-CRC polynomial used in many protocols

When a station needs to transmit, it waits until the LAN becomes silent for a specified period (interframe gap). When a station starts sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a frame. The station then waits a random time period (backoff) before attempting to send again. When the backoff completes, the station waits for silence on the LAN and begins retransmission on the LAN. This process is called a retry. If the frame is not successfully sent within 15 retries, an error is indicated.

10-Mbps Ethernet basic timing specifications follow:

- Transmits at 0.8 µs per byte
- The preamble plus start frame delimiter is sent in 6.4 µs.
- The minimum interframe gap is 9.6 µs.
- The slot time is 51.2 µs.

100-Mbps Ethernet basic timing specifications follow:

- Transmits at 0.08 µs per byte
- The preamble plus start frame delimiter is sent in 0.64 µs.
- The minimum interframe gap is 0.96 µs.
- The slot time is 5.12 µs.

32.1 Fast Ethernet on the MPC8272

When a general FCC mode register (GFMR*x*[MODE]) selects Ethernet protocol, that FCC performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control (MAC) and channel interface functions. Figure 32-2 shows a block diagram of the FCC Ethernet control logic.

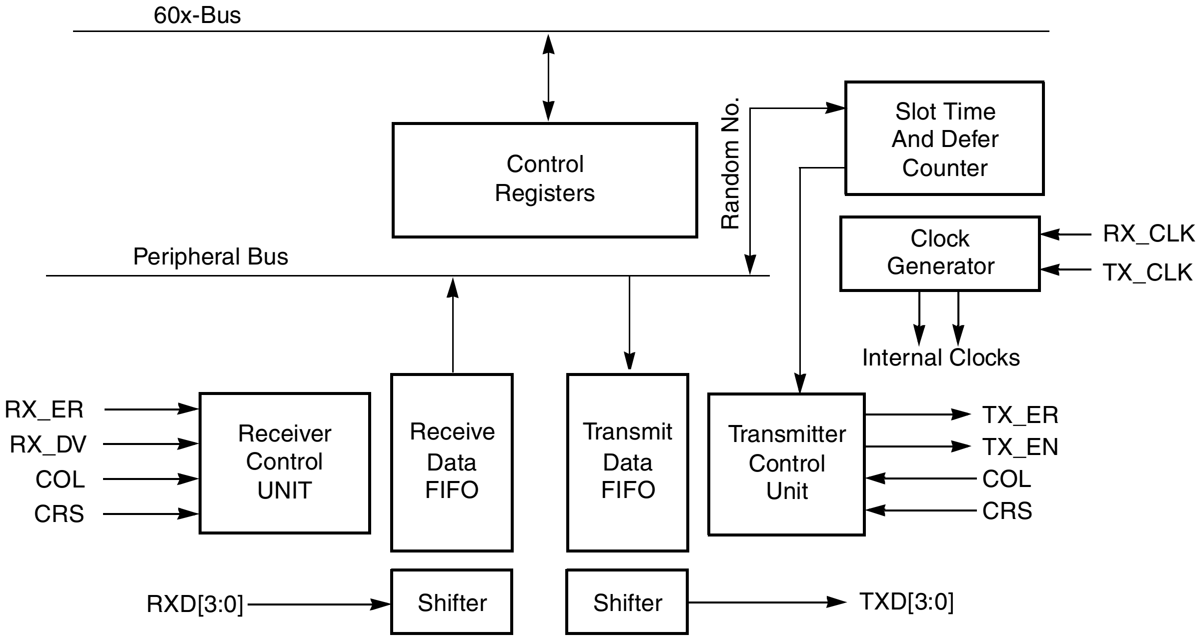


Figure 32-2. Ethernet Block Diagram

32.2 Features

The following is a list of fast Ethernet key features:

- Support for fast Ethernet through the MII (media-independent interface) and the RMII (reduced media-independent interface)
- Performs MAC (media access control) layer functions of fast Ethernet and IEEE 802.3x
- Performs framing functions

- Preamble generation and stripping
- Destination address checking
- CRC generation and checking
- Automatic padding of short frames on transmit
- Framing error (dribbling bits) handling

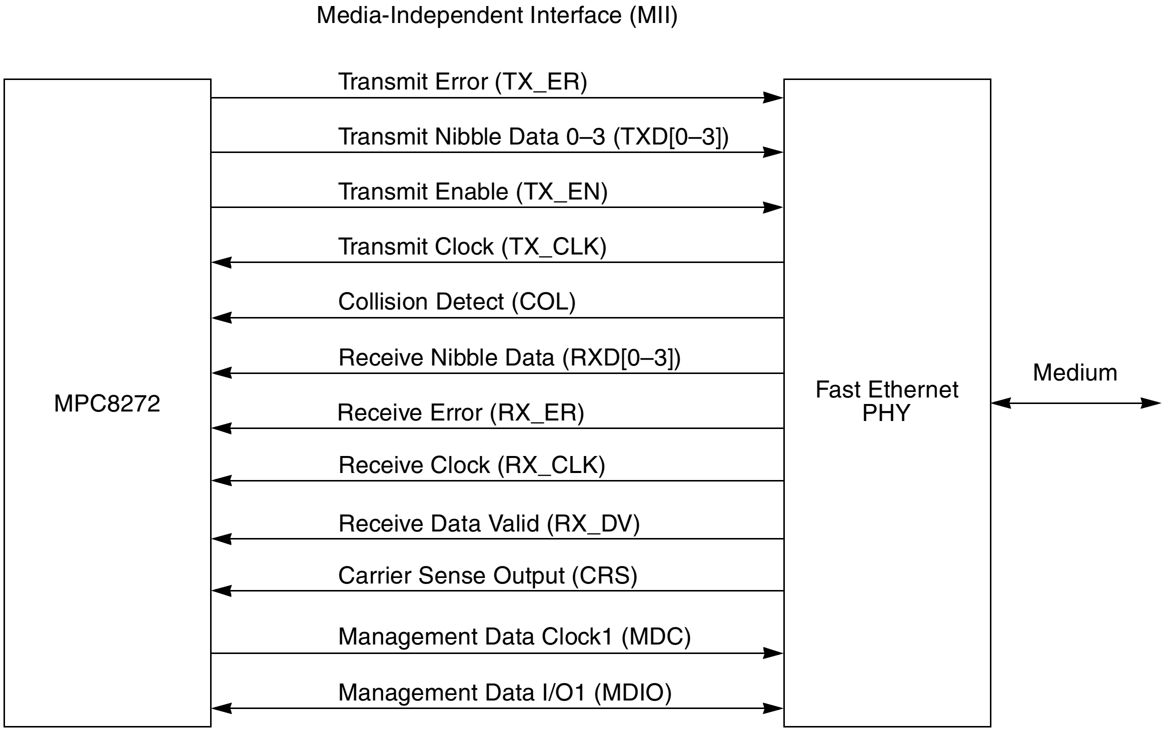
- Full collision support
  - Enforces the collision (jamming and TX_ER assertion)
  - Truncated binary exponential backoff algorithm for random wait
  - Two nonaggressive backoff modes
  - Automatic frame retransmission (until retry limit is reached)
  - Automatic discard of incoming collided frames
  - Delay transmission of new frames for specified interframe gap
- Bit rates up to 100 Mbps
- Receives back-to-back frames
- Detection of receive frames that are too long
- Multibuffer data structure
- Supports 48-bit addresses in three modes
  - Physical. One 48-bit address recognized or 64-bin hash table for physical addresses.
  - Logical. 64-bin group address hash table plus broadcast address checking.
  - Promiscuous. Receives all frames regardless of address (a CAM can be used for address filtering).
- External CAM support on system bus interfaces
- Special RMON counters for monitoring network statistics
- Transmitter network management and diagnostics
  - Lost carrier sense
  - Underrun
  - Number of collisions exceeded the maximum allowed
  - Number of retries per frame
  - Deferred frame indication
  - Late collision
- Receiver network management and diagnostics
  - CRC error indication
  - Nonoctet alignment error
  - Frame too short
  - Frame too long
  - Overrun
  - Busy (out of buffers)

- Error counters
  - — Discarded frames (out of buffers or overrun occurred)
  - — CRC errors
  - — Alignment errors
- Internal and external loopback mode
- Supports fast Ethernet in duplex mode
- Supports pause flow control frames
- Support of out-of-sequence transmit queue (for flow-control frames)
- External buffer descriptors (BDs)

32.3 Connecting the MPC8272 to Fast Ethernet

32.3.1 Connecting the MPC8272 to Ethernet (MII)

Figure 32-3 shows the basic components of the media-independent interface (MII) and the signals required to make the fast Ethernet connection between the MPC8272 and a PHY.


Media-Independent Interface (MII)
MPC8272
Transmit Error (TX_ER)
Transmit Nibble Data 0–3 (TXD[0–3])
Transmit Enable (TX_EN)
Transmit Clock (TX_CLK)
Collision Detect (COL)
Receive Nibble Data (RXD[0–3])
Receive Error (RX_ER)
Receive Clock (RX_CLK)
Receive Data Valid (RX_DV)
Carrier Sense Output (CRS)
Management Data Clock1 (MDC)
Management Data I/O1 (MDIO)
Fast Ethernet PHY
Medium

[1] The management signals (MDC and MDIO) can be common to all of the fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 32-3. Connecting the MPC8272 to Ethernet

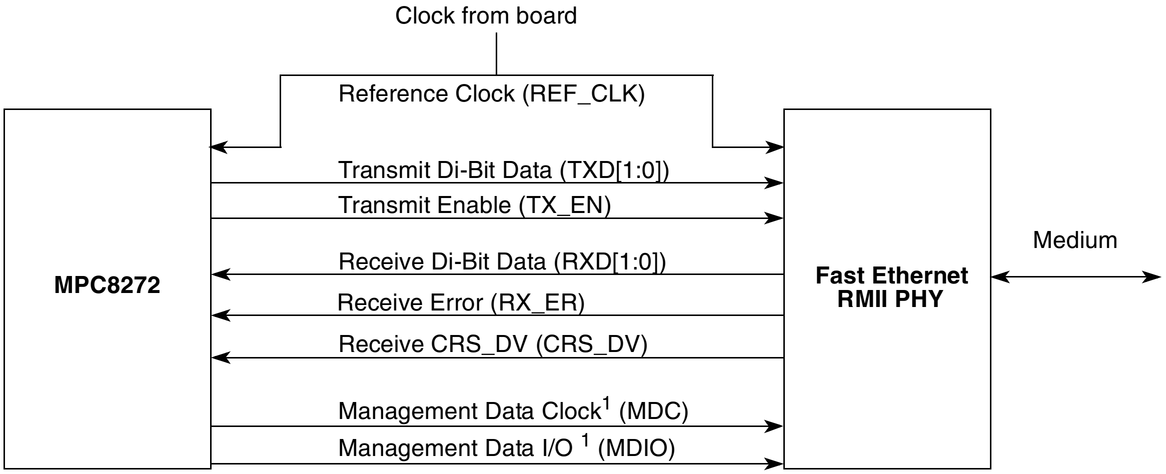
Each FCC has 18 signals, defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY. The two management signals (MDC and MDIO) required by the MII should be implemented separately using the parallel I/O.

The MPC8272 has additional signals for interfacing with an optional external content-addressable memory (CAM), which are described in Section 32.7, “CAM Interface.”

The MPC8272 uses the SDMA channels to store every byte received after the start frame delimiter into system memory. On transmit, the user provides the destination address, source address, type/length field, and transmit data. To meet minimum frame requirements, MPC8272 automatically pads frames with fewer than 64 bytes in the data field. The MPC8272 also appends the FCS to the frame.

32.3.2 Connecting the MPC8272 to Ethernet (RMII)

Figure 32-4 shows the basic components of the reduced media-independent interface (RMII) and the signals required for the fast Ethernet connection between the MPC8272 and a PHY. The MDC/MDIO management interface is the same as in MII. The RMII reference clock (REF_CLK) is distributed over the FCC transmit clock. In RMII mode receive clock is not used.


Clock from board
Reference Clock (REF_CLK)
Transmit Di-Bit Data (TXD[1:0])
Transmit Enable (TX_EN)
Receive Di-Bit Data (RXD[1:0])
Receive Error (RX_ER)
Receive CRS_DV (CRS_DV)
Management Data Clock[1] (MDC)
Management Data I/O [1] (MDIO)
MPC8272
Fast Ethernet RMII PHY
Medium

[1] The management signals (MDC and MDIO) can be common to all of the fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 32-4. Connecting the MPC8272 to Ethernet (RMII)

32.4 Ethernet Channel Frame Transmission

The Ethernet transmitter requires almost no core intervention. When the core enables the transmitter, the Ethernet controller polls the first TxBD in the FCC’s TxBD table every 256 serial clocks. If the user has a frame ready to transmit, setting FTODR[TOD] eliminates waiting for the next poll. When there is a frame to transmit, the Ethernet controller begins fetching the data from the data buffer and asserts TX_EN. The preamble sequence, start frame delimiter, and frame information are sent in that order; see Figure 32-1. In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap 28 serial clocks (112 bit time period) regardless of CRS assertion.

There is one internal buffer for out-of-sequence flow control frames (in full-duplex fast Ethernet). When the fast Ethernet controller is between frames, this buffer is polled if flow control is enabled. This buffer must contain the whole frame.

However, in half-duplex mode, the controller defers transmission if the line is busy (CRS asserted). Before transmitting, the controller waits for carrier sense to become inactive, at which point the controller determines if CRS remains negated for 16 serial clocks. If so, the transmission begins after an additional 8 serial clocks (96 bit times after CRS originally became negated). In the fast Ethernet transmitter, if CRS is asserted and then negated within 10 clocks after TXEN is negated, the next frame is not deferred and a defer indication is asserted.

If a collision occurs during the transmit frame, the Ethernet controller follows a specified backoff procedure and tries to retransmit the frame until the retry limit is reached. The Ethernet controller stores at least the first 64 bytes of data of the transmit frame in the FCC FIFO, so that the data does not have to be retrieved from system memory in case of a collision. This improves bus usage and latency if the backoff timer output requires an immediate retransmission.

When the end of the current buffer is reached and TxBD[L] = 1, the FCS (32-bit CRC) bytes are appended (if TxBD[TC] = 1), and TX_EN is negated. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. When the end of the current buffer is reached and TxBD[L] = 0 (a frame is comprised of multiple buffers), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD] = 1, the Ethernet controller pads short frames to the value of the minimum frame length register (MINFLR), described in Table 32-2.

To rearrange the transmit queue before the CP finishes sending all frames, issue a GRACEFUL STOP TRANSMIT command. This can be useful for transmitting expedited data ahead of previously linked buffers or for error situations. When the GRACEFUL STOP TRANSMIT command is issued, the Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with a collision. When the Ethernet controller is given the RESTART TRANSMIT command, it resumes transmission. The Ethernet controller sends bytes least-significant nibble first.

32.5 Ethernet Channel Frame Reception

The Ethernet receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, short frame checking, maximum DMA transfer checking, and maximum frame-length checking.

When the core enables the Ethernet receiver, it enters hunt mode when RX_DV is asserted as long as COL remains negated (full-duplex mode ignores COL). In hunt mode, as data is shifted into the receive shift register 4 bits at a time, the contents of the register are compared to the contents of the SYN2 field in the FCC’s data synchronization register (FDSR). When the registers match, the hunt mode is terminated and character assembly begins.

When the receiver detects the first bytes of a frame, the Ethernet controller performs address recognition functions on the frame; see Section 32.12, "Ethernet Address Recognition." The receiver can receive physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal address recognition algorithm is complete, bus usage is not wasted on frames not addressed to this station. The receiver can also operate with an external CAM, in which case frame reception continues normally, unless the CAM specifically signals the frame to be rejected. See Section 32.7, "CAM Interface."

If an address is recognized, the Ethernet controller fetches the next RxBD and, if it is empty, starts transferring the incoming frame to the RxBD's associated data buffer.

In half-duplex mode, if a collision is detected during the frame, the RxBDs associated with this frame are reused. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems. When the buffer has been filled, the Ethernet controller clears RxBD[E] and generates an interrupt if RxBD[I] is set. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table; if it is empty, it continues receiving the rest of the frame.

The RxBD length is determined by MRBLR in the parameter RAM. The user should program MRBLR to be at least 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. When the frame ends, the receive CRC field is checked and written to the data buffer. The data length written to the last BD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

If an external CAM is used (FPSMR*x*[CAM] = 1), the Ethernet controller adds the two lower bytes of the CAM output at the end of each frame. Note that the data length does not include these two bytes; that is, the extra two bytes could push the buffer length past MRBLR.

When the receive frame is complete, the Ethernet controller sets RxBD[L], writes the other frame status bits into the RxBD, and clears RxBD[E]. The Ethernet controller next generates a maskable interrupt, indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame. The Ethernet controller receives serial data least-significant nibble first.

32.6 Flow Control

Because collisions cannot occur in full-duplex mode, fast Ethernet can operate at the maximum rate. When the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 32-1 shows the flow-control frame structure.

Table 32-1. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address		
2	Length/type	88-08	Control frame type

Table 32-1. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	MAC opcode	00-01	Pause command
2	MAC parameter	Up to 0xFFFE	Pause period measured in slot times, most-significant octet first with a two time-slot resolution. Note: Because the pause period has a resolution of two time slots, the value programmed in this field is rounded up to the nearest even number before being used, as follows: MAC Parameter ValuePause Period 0 none 1 or 2 2 x slot time 3 or 4 4 x slot time … …
42	Reserved	—	
4	FCS		Frame check sequence (CRC)

When flow-control mode is enabled (FPSMR*x*[FCE]) and the receiver identifies a pause-flow control frame sent to individual or broadcast addresses, transmission stops for the time specified in the control frame. During this pause, only the out-of-sequence frame is sent. Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

32.7 CAM Interface

The MPC8272 internal address recognition logic can be used in combination with an external CAM. When using a CAM, the FCC must be in promiscuous mode (FPSMR*x*[PRO] = 1). See Section 32.12, “Ethernet Address Recognition.”

The Ethernet controller writes two 32-bit accesses to the CAM and reads the result in a 32-bit access. If the high bit of the result is set, the frame is rejected; otherwise, the lower 16 bits are attached to the end of the frame.

When an external CAM is used for address filtering, users can choose to either discard rejected frames (FPSMR[ECM] = 0) or receive rejected frames and signal the CAM miss in the RxBD (FPSMR[ECM] = 1).

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

32.8 Ethernet Parameter RAM

For Ethernet mode, the protocol-specific area of the FCC parameter RAM is mapped as in Table 32-2.

Table 32-2. Ethernet-Specific Parameter RAM

Offset[1]	Name	Width	Description
0x3C	STAT_BUF	Word	Buffer of internal usage
0x40	**CAM_PTR**	Word	CAM address. For FCC fast Ethernet operation the CAM should be located on the same bus as the data buffers.
0x44	**C_MASK**	Word	Constant MASK for CRC (initialize to 0xDEBB_20E3). For the 32-bit CRC-CCITT.
0x48	**C_PRES**	Word	Preset CRC (initialize to 0xFFFF_FFFF). For the 32-bit CRC-CCITT.
0x4C	**CRCEC**[2]	Word	CRC error counter. Counts each received frame with a CRC error. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, frames with overrun errors, or frames with alignment errors.
0x50	**ALEC** [2]	Word	Alignment error counter. Counts frames received with dribbling bits. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, or frames with overrun errors.
0x54	**DISFC** [2]	Word	Discard frame counter. Incremented for discarded frames because of an out-of-buffers condition or overrun error. The CRC need not be correct for this counter to be incremented.
0x58	**RET_LIM**	Hword	Retry limit (typically 15 decimal). Number of retries that should be made to send a frame. If the frame is not sent after this limit is reached, an interrupt can be generated.
0x5A	RET_CNT	Hword	Retry limit counter. Temporary decrementer used to count retries made.
0x5C	**P_PER**	Hword	Persistence. Allows the Ethernet controller to be less persistent after a collision. Normally cleared, P_PER can be from 0 to 9 (9 = least persistent). The value is added to the retry count in the backoff algorithm to reduce the chance of transmission on the next time-slot. Using a less persistent backoff algorithm increases throughput in a congested Ethernet LAN by reducing the chance of collisions. FPSMR[SBT] can also reduce persistence of the Ethernet controller. The Ethernet/802.3 specifications permit the use of P_PER.
0x5E	BOFF_CNT	Hword	Backoff counter
0x60	**GADDR_H**	Word	Group address filters high and low are used in the hash table function of the group addressing mode. The user may write zeros to these values after reset and before the Ethernet channel is enabled to disable all group hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table. See Section 32.13, “Hash Table Algorithm.”
0x64	**GADDR_L**	Word	
0x68	**TFCSTAT**	Hword	Out-of-sequence TxBD. Includes the status/control, data length, and buffer pointer fields in the same format as a regular TxBD. Useful for sending flow control frames. This area’s TxBD[R] is always checked between frames, regardless of FPSMR*x*[FCE]. If it is not ready, a regular frame is sent. The user must set TxBD[L] when preparing this BD. If TxBD[I] is set, a TXC event is generated after frame transmission. This area should be cleared when not in use.
0x6A	**TFCLEN**	Hword	
0x6C	**TFCPTR**	Word	
0x70	**MFLR**	Hword	Maximum frame length register (typically1518 decimal). If the Ethernet controller detects an incoming frame exceeding MFLR, it sets RxBD[LG] (frame too long) in the last RxBD, but does not discard the rest of the frame. The controller also reports the frame status and length of the received frame in the last RxBD. MFLR includes all in-frame bytes between the start frame delimiter and the end of the frame.

Table 32-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name	Width	Description
0x72	**PADDR1_H**	Hword	The 48-bit individual address of this station. PADDR1_L is the lowest order half-word, and PADDR1_H is the highest order half-word.
0x74	**PADDR1_M**	Hword	
0x76	**PADDR1_L**	Hword	
0x78	IBD_CNT	Hword	Internal BD counter
0x7A	IBD_START	Hword	Internal BD start pointer
0x7C	IBD_END	Hword	Internal BD end pointer
0x7E	TX_LEN	Hword	Tx frame length counter
0x80	IBD_BASE	32 Bytes	Internal microcode usage
0xA0	**IADDR_H**	Word	Individual address filter high/low. Used in the hash table function of the individual addressing mode. The user can write zeros to these values after reset and before the Ethernet channel is enabled to disable all individual hash address recognition functions. Issuing a SET GROUP ADDRESS command enables the hash table. See Section 32.13, “Hash Table Algorithm.”
0xA4	**IADDR_L**	Word	
0xA8	**MINFLR**	Hword	Minimum frame length register (typically 64 decimal). If the Ethernet receiver detects an incoming frame shorter than MINFLR, it discards that frame unless FPSMR[RSH] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The Ethernet transmitter pads frames that are too short (according to TxBD[PAD] and the PAD value in the parameter RAM). PADs are added to make the transmit frame MINFLR bytes.
0xAA	**TADDR_H**	Hword	Allows addition of addresses to the individual and group hashing tables. After an address is placed in TADDR, issue a SET GROUP ADDRESS command. TADDR_L is the lowest-order half-word; TADDR_H is the highest. A zero in the I/G bit indicates an individual address; 1 indicates a group address.
0xAC	**TADDR_M**	Hword	
0xAE	**TADDR_L**	Hword	
0xB0	**PAD_PTR**	Hword	Internal PAD pointer. This internal 32-byte aligned pointer points to a 32-byte buffer filled with pad characters. The pads may be any value, but all the bytes should be the same to assure padding with a specific character. If a specific padding character is not needed, PAD_PTR should equal the internal temporary data pointer TIPTR; see Section 29.7, “FCC Parameter RAM.”
0xB2	—	Hword	Reserved, should be cleared
0xB4	CF_RANGE	Hword	Control frame range. Internal usage.
0xB6	MAX_B	Hword	Maximum BD byte count. Internal usage.
0xB8	**MAXD1**	Hword	Max DMA1 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The MAXD1 value is valid only if an address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the user-defined value in MAXD1, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes have been received) and reports the frame status and length (including the discarded bytes) in the last RxBD. This value must be greater than 32.

Table 32-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name	Width	Description
0xBA	**MAXD2**	Hword	Max DMA2 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The value of MAXD2 is valid in promiscuous mode when no address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the value in MAXD2, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes are received) and reports frame status and length (including the discarded bytes) in the last RxBD. In a monitor station, MAXD2 can be much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames.This value must be less than MAXD1.
0xBC	MAXD	Hword	Rx maximum DMA. Internal usage
0xBE	DMA_CNT	Hword	Rx DMA counter. Temporary down-counter used to track the frame length.
0xC0	**OCTC** [2]	Word	(RMON mode only) The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).
0xC4	**COLC** [2]	Word	(RMON mode only) The best estimate of the total number of collisions on this Ethernet segment.
0xC8	**BROC** [2]	Word	(RMON mode only) The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.
0xCC	**MULC** [2]	Word	(RMON mode only) The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.
0xD0	**USPC** [2]	Word	(RMON mode only) The total number of packets received that were less than 64 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xD4	**FRGC** [2]	Word	(RMON mode only) The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment because it counts both runts (which are normal occurrences due to collisions) and noise hits.
0xD8	**OSPC** [2]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xDC	**JBRC** [2]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets), and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3, Section 8.2.1.5 (10BASE5) and Section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.
0xE0	**P64C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).
0xE4	**P65C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).

Table 32-2. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name	Width	Description
0xE8	**P128C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).
0xEC	**P256C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).
0xF0	**P512C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).
0xF4	**P1024C** [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets).
0xF8	CAM_BUF	Word	Internal buffer for CAM result
0xFC	—	Word	Reserved, should be cleared
0xFF	—	Byte	10 Mbps Poll Delay 100 Mbps = 0x0 10 Mbps = 0x0B (up to 100 Mhz CPM) = 0x18 (up to 200 Mhz CPM) = 0x32 (up to 333 Mhz CPM)

[1] Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 13.5.2, “Parameter RAM.”

[2] 32-bit (modulo 232) counters maintained by the CP; cleared by the user while the channel is disabled.

32.9 Programming Model

The core configures an FCC to operate as an Ethernet controller using GFMR[MODE]. The receive errors (collision, overrun, nonoctet-aligned frame, short frame, frame too long, and CRC error) are reported through the RxBD. The transmit errors (underrun, heartbeat, late collision, retransmission limit, and carrier sense lost) are reported through the TxBD.

The user should program the FDSR as described in Section 29.4, “FCC Data Synchronization Registers (FDSRx),” with FDSR[SYN2] = 0xD5 and FDSR[SYN1] = 0x55.

32.10 Ethernet Command Set

The transmit and receive commands are issued to the CPCR; see Section 13.4, “Command Set.”

NOTE

Before resetting the CPM, configure TX_EN ($\overline{\text{RTS}}$) to be an input.

Transmit commands that apply to Ethernet are described in Table 32-3.

Table 32-3. Transmit Commands

Command	Description
STOP TRANSMIT	When used with the Ethernet controller, this command violates a specific behavior of an Ethernet/IEEE 802.3 station. It should not be used.
GRACEFUL STOP TRANSMIT	Used to smoothly stop transmission after the current frame finishes sending or undergoes a collision (immediately if there is no frame being sent). FCCE[GRA] is set once transmission stops. Then the Ethernet transmit parameters (including BDs) can be modified by the user. The TBPTR points to the next TxBD in the table. Transmission begins when the R bit of the next BD is set and the RESTART TRANSMIT command is issued. Note that if the GRACEFUL STOP TRANSMIT command is issued and the current transmit frame ends in a collision, the TBPTR points to the beginning of the collided frame with TxBD[R] still set (the frame looks as if it was never sent).
RESTART TRANSMIT	Enables transmission of characters on the transmit channel. It is expected by the Ethernet controller after a GRACEFUL STOP TRANSMIT command or transmitter error (underrun, retransmission limit reached, or late collision). The Ethernet controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all the transmit parameters in this serial channel parameter RAM to their reset state. This command should be issued only when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Receive commands that apply to Ethernet are described in Table 32-4.

Table 32-4. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel in the FCC mode register is enabled, the channel is in the receive enable mode and uses the first BD in the table. This command is generally used to force the Ethernet receiver to abort reception of the current frame and enter hunt mode. In hunt mode, the Ethernet controller continually scans the input data stream for a transition of carrier sense from inactive to active followed by a preamble sequence and the start frame delimiter. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD. Note that short frames pending in the internal FIFO may be lost.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the receive and transmit parameters.
SET GROUP ADDRESS	Used to set one of the 64 bits of the four individual/group address hash filter registers (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.

If an address from the hash table must be deleted, the Ethernet channel must be disabled, the hash table registers must be cleared, and the SET GROUP ADDRESS command must be executed for the remaining preferred addresses. This is required because the hash table might have mapped multiple addresses to the same hash table bit.

32.11 RMON Support

The fast Ethernet controller can automatically gather network statistics required for RMON without the need to receive all addresses using promiscuous mode. Setting FPSMR*x*[MON] enables RMON support.

The RMON statistics and their corresponding counters in the parameter RAM are described in Table 32-5.

Table 32-5. RMON Statistics and Counters

Statistic	Description	Counter
etherStatsDropEvents	The total number of events in which packets were detected as dropped by the probe due to lack of resources. Note that this may not be the number of packets dropped; it is the number of times this condition is detected.	DISFC
etherStatsOctets	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).	OCTC
etherStatsPkts	The total number of packets (including bad packets, broadcast packets, and multicast packets) received.	USPC + OSPC + FRGC + JBRC + P64C + P65C + P128C + P256C + P512C + P1024C
etherStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.	BROC
etherStatsMulticastPkts	The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.	MULC
etherStatsCRCAlignErrors	The total number of packets received that had a length (excluding framing bits but including FCS octets) of between 64 and 1518 octets, inclusive, but had either an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error).	CRCEC + ALEC -FRGC -JBRC
etherStatsUndersizePkts	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and were otherwise well-formed.	USPC
etherStatsOversizePkts	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.	OSPC
etherStatsFragments	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment, because it counts both runts (which are normal occurrences due to collisions) and noise hits.	FRGC

Table 32-5. RMON Statistics and Counters (continued)

Statistic	Description	Counter
etherStatsJabbers	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3, Section 8.2.1.5 (10BASE5) and Section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.	JBRC
etherStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment	COLC
etherStatsPkts64Octets	The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).	P64C
etherStatsPkts65to127Octets	The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).	P65C
etherStatsPkts128to255Octets	The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).	P128C
etherStatsPkts256to511Octets	The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).	P256C
etherStatsPkts512to1023Octets	The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).	P512C
etherStatsPkts1024to1518Octets	The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets)	P1024C

32.12 Ethernet Address Recognition

The Ethernet controller can filter the received frames based on different addressing types—physical (individual), group (multicast), broadcast (all-ones group address), and promiscuous. The difference between an individual address and a group address is determined by the I/G bit in the destination address field.

Figure 32-5 is a flowchart for address recognition on received frames.

Do not

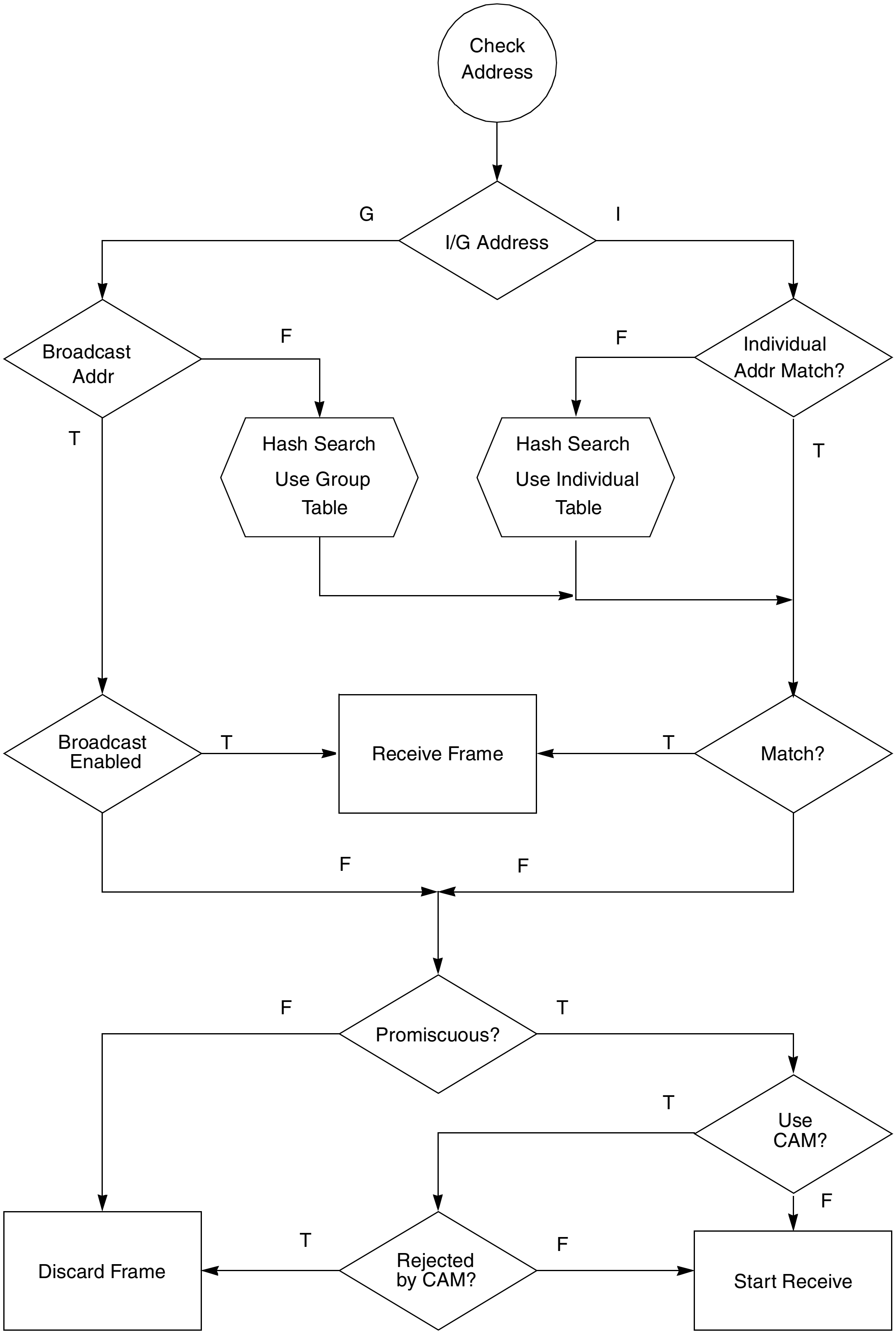

Check Address
I/G Address
G
I
Broadcast Addr
F
T
Individual Addr Match?
F
T
Hash Search Use Group Table
Hash Search Use Individual Table
Broadcast Enabled
T
Receive Frame
T
Match?
F
F
Promiscuous?
F
T
Use CAM?
T
F
Discard Frame
T
Rejected by CAM?
F
Start Receive

Figure 32-5. Ethernet Address Recognition Flowchart

In the physical type of address recognition, the Ethernet controller compares the destination address field of the received frame with the physical address that the user programs in the PADDR. If it fails, the controller performs address recognition on multiple individual addresses using the IADDR_H/L hash table. Since the controller always checks PADDR and the individual hash, for individual address the user must write zeros to the hash in order to avoid a hash match and ones to PADDR in order to avoid individual address match.

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast and broadcast addresses are enabled, the frame is accepted. If the group address is not a broadcast address, the user can perform address recognition on multiple group addresses using the GADDR_H/L hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address when an external CAM is not used.

If an external CAM is used for address recognition (FPSMR[CAM] = 1), the user should select promiscuous mode; the frame can be rejected if there is no match in the CAM. If the on-chip address recognition functions detect a match, the external CAM is not accessed. Otherwise, the CPM issues a match transaction to the CAM using the bus on which the data buffers reside. (The data buffer bus is selected in FCR*x*[DTB]; see Section 29.7.1, “FCC Function Code Registers (FCRx).”)

32.13 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit address into one of 64 bins, which are represented by the 64 bits in GADDR_H/L or IADDR_H/L. When the SET GROUP ADDRESS command is executed, the Ethernet controller maps the selected 48-bit address in TADDR into one of the 64 bits. This is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and using 6 bits of the CRC-encoded result to generate a number between 1 and 64. Bit 26 of the CRC result selects which address filter registers are used in the hashing process—either GADDR_H/IADDR_H or GADDR_L/IADDR_L— and bits 27–31 of the CRC result select which bit is set.

The same process is used when the Ethernet controller receives a frame. If the CRC generator selects a bit that is set in the group/individual hash table, the frame is accepted; otherwise, it is rejected. The result is that if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. The core must further filter those that reach memory to determine if they contain one of the eight preferred addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if eight group and eight physical addresses are stored in their respective hash tables, 87.5% of all frames (not just group address frames) are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, with 128 addresses stored in a 64-bin hash table, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory. In such instances, an external CAM is advised if the extra bus use cannot be tolerated. See Section 32.7, “CAM Interface.”

NOTE

The hash tables cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. Thus, an external CAM must be used to implement this function.

32.14 Interpacket Gap Time

The minimum interpacket gap time for back-to-back transmission is 96 bit times. The receiver receives back-to-back frames with this minimum spacing. In addition, after the backoff algorithm, the transmitter waits for carrier sense to be negated before retransmitting the frame. The retransmission begins 96 bit times after carrier sense is negated if it stays negated for at least 64 bit times. So if there is no change in the carrier sense indication during the first 64 bit times (16 serial clocks), the retransmission begins 96 clocks after carrier sense is first negated

32.15 Handling Collisions

If a collision occurs during frame transmission, the Ethernet controller continues transmission for at least 32 bit times, transmitting a jam pattern of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the sequence ends.

If a collision occurs within 64 byte times, the process is retried. The transmitter waits a random number of slot times. (A slot time is 512 bit times.) If a collision occurs after 64 byte times, no retransmission is performed, FCCE[TXE] is set, and the buffer is closed with a late-collision error indication in TxBD[LC]. If a collision occurs during frame reception, reception is stopped. This error is reported only in the RxBD if the frame is at least as long as the MINFLR or if FPSMR[RSH] = 1.

32.16 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both receive and transmit FIFO buffers are used and the FCC operates in full duplex. Both internal and external loopback are configured using combinations of FPSMR[LPB] and GFMR[DIAG]. Because of the full-duplex nature of the loopback operation, the performance of the other FCCs is degraded.

Internal loopback disconnects the FCC from the SI. The receive data is connected to the transmit data. The transmitted data from the transmit FIFO is received immediately into the receive FIFO. There is no heartbeat check in this mode.

In external loopback operation, the Ethernet controller listens for data received from the PHY while it is sending.

32.17 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the FCC event register.

Transmission errors are described in Table 32-6.

Table 32-6. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, closes the buffer, sets TxBD[UN] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command.
Carrier sense lost during frame transmission	If no collision is detected in the frame, the controller sets TxBD[CSL] and FCCE[TXE], and it continues the buffer transmission normally. No retries are performed as a result of this error.
Retransmission attempts limit expired	The controller terminates buffer transmission, closes the buffer, sets TxBD[RL] and FCCE[TXE]. Transmission resumes after receiving the RESTART TRANSMIT command.
Late collision	The controller terminates buffer transmission, closes the buffer, sets TxBD[LC] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command. Note that late collision parameters are defined in FPSMR[LCW].

Reception errors are described in Table 32-7.

Table 32-7. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller writes the received data byte to the internal FIFO buffer over the previously received byte. The previous data byte and frame status are lost. The controller closes the buffer, sets RxBD[OV] and FCCE[RXF], and increments the discarded frame counter (DISFC). The receiver then enters hunt mode.
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets FCCE[BSY] and increments the discarded frame counter (DISFC).
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits when the receive frame terminates as nonoctet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame nonoctet aligned (RxBD[NO]) error is reported, FCCE[RXF] is set, and the alignment error counter (ALEC) in the parameter RAM is incremented. If there is no CRC error, no error is reported.
CRC error	When a CRC error occurs, the controller closes the buffer, and sets RxBD[CR] and FCCE[RXF]. Also, the CRC error counter (CRCEC) in the parameter RAM is incremented. After receiving a frame with a CRC error, the receiver enters hunt mode. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

32.18 Fast Ethernet Registers

The following sections describe registers used for configuring and operating the fast Ethernet controller.

32.18.1 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3 7
Field	TIREM	LPB	CLK	—
Reset	0000_0000			
R/W	R/W			
Addr	0x11390 (GFEMR1), 0x113B0(GFEMR2)			

Figure 32-6. General FCC Expansion Mode Register (GFEMR)

Table 32-8 describes GFEMR*x* fields.

Table 32-8. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0–3] is controlled only by FTIRR[0–3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0–31] are controlled by FTIRR[0–3], FIRPER, FIRSR_HI and FIRSR_LO. Underrun status for PHYs[0–31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII loopback diagnostic mode (Ethernet mode): 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50-MHz input clock from external oscillator (Ethernet mode): 0 50 MHz (for fast Ethernet) 1 5 MHz (for 10BaseT)
3–7	—	Reserved, should be cleared.

32.18.2 FCC Ethernet Mode Register (FPSMR)

The MPC8272 supports 10/100 Mbps Ethernet through a RMII interface (according to RMII Specification March 20, 1998). The RMII use a single reference clock (50 MHz) and seven pins that are a proper subset of the MII interface pins. Ethernet features are unchanged in RMII mode. To select RMII-PHY interface, a mode bit in the Ethernet mode register (FPSMR) has been added, as shown in Figure 32-7.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	HBC	FC	SBT	LPB	LCW	FDE	MON	—		PRO	FCE	RSH	—		RMII	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11304 (FPSMR1), 0x11324 (FPSMR2)															

	16	20	21	22	23	24	25	26	31
Field	—		CAM	BRO	—	CRC		—	
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x11306 (FPSMR1), 0x11326 (FPSMR2)								

Figure 32-7. FCC Ethernet Mode Register (FPSMR*x*)

Table 32-9 describes FPSMR fields.

Table 32-9. FPSMR Ethernet Field Descriptions

Bits	Name	Description
0	HBC	Heartbeat checking 0 No heartbeat checking is performed. Do not wait for a collision after transmission. 1 Wait 40 transmit serial clocks for a collision asserted by the transceiver after transmission. TxBD[HB] is set if the heartbeat is not heard within 40 transmit serial clocks.
1	FC	Force collision 0 Normal operation 1 The channel forces a collision on transmission of every transmit frame. The MPC8272 should be configured in loopback operation when using this feature, which allows the user to test the MPC8272 collision logic. It causes the retry limit to be exceeded for each transmit frame.
2	SBT	Stop backoff timer 0 The backoff timer functions normally. 1 The backoff timer (for the random wait after a collision) is stopped whenever carrier sense is active. In this method, the retransmission is less aggressive than the maximum allowed in the IEEE 802.3 standard. The persistence (P_PER) feature in the parameter RAM can be used in combination with the SBT bit (or in place of the SBT bit).
3	LPB	Local protect bit. 0 Receiver is blocked when transmitter sends (default). 1 Receiver is not blocked when transmitter sends. Must be set for full-duplex operation. For loopback operation, GFMR[DIAG] must be programmed also; see Section 29.2.1, “General FCC Mode Registers (GFMRx)."
4	LCW	Late collision window 0 A late collision is any collision that occurs at least 64 bytes from the preamble. 1 A late collision is any collision that occurs at least 56 bytes from the preamble.
5	FDE	Full duplex Ethernet 0 Disable full-duplex 1 Enable full-duplex. Must be set if FSMR[LPB] is set or external loopback is performed.
6	MON	RMON mode 0 Disable RMON mode 1 Enable RMON mode

Table 32-9. FPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
7–8	—	Reserved, should be zero
9	PRO	Promiscuous 0 Check the destination address of incoming frames. 1 Receive the frame regardless of its address. A CAM can be used for address filtering when FSMR[CAM] is set.
10	FCE	Flow control enable 0 Flow control is not enabled 1 Flow control is enabled
11	RSH	Receive short frames 0 Discard short frames (frames smaller than the value specified in MINFLR). 1 Receive short frames.
12–13	—	Reserved, should be zero.
14	RMII	RMII interface mode 0 MII interface 1 RMII interface. RMII to/from MII conversion logic is enabled.
15–20	—	Reserved, should be zero.
21	CAM	CAM address matching 0 Normal operation. 1 Use the CAM for address matching; CAM result (16 bits) is added at the end of the frame.
22	BRO	Broadcast address 0 Receive all frames containing the broadcast address. 1 Reject all frames containing the broadcast address unless FSMR[PRO] = 1.
23	—	Reserved, should be zero
24–25	CRC	CRC selection 0x Reserved. 10 32-bit CCITT-CRC (Ethernet). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1. Select this to comply with Ethernet specifications. 11 Reserved.
26–31	—	Reserved, should be zero

32.18.3 Ethernet Event Register (FCCE)/Mask Register (FCCM)

The FCCE, shown in Figure 32-8, is used as the Ethernet event register when the FCC functions as an Ethernet controller. It generates interrupts and reports events recognized by the Ethernet channel. On recognition of an event, the Ethernet controller sets the corresponding FCCE bit. Interrupts generated by this register can be masked in the Ethernet mask register (FCCM).

The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

The FCCE can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values. Unmasked FCCE bits must be cleared before the CP clears the internal interrupt request.

Figure 32-8. Ethernet Event Register (FCCE)/Mask Register (FCCM)

	0 7	8	9	10	11	12	13	14	15
Field	—	GRA	RXC	TXC	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x11310 (FCCE1), 0x11330 (FCCE2), 0x11314 (FCCM1), 0x11334 (FCCM2)								

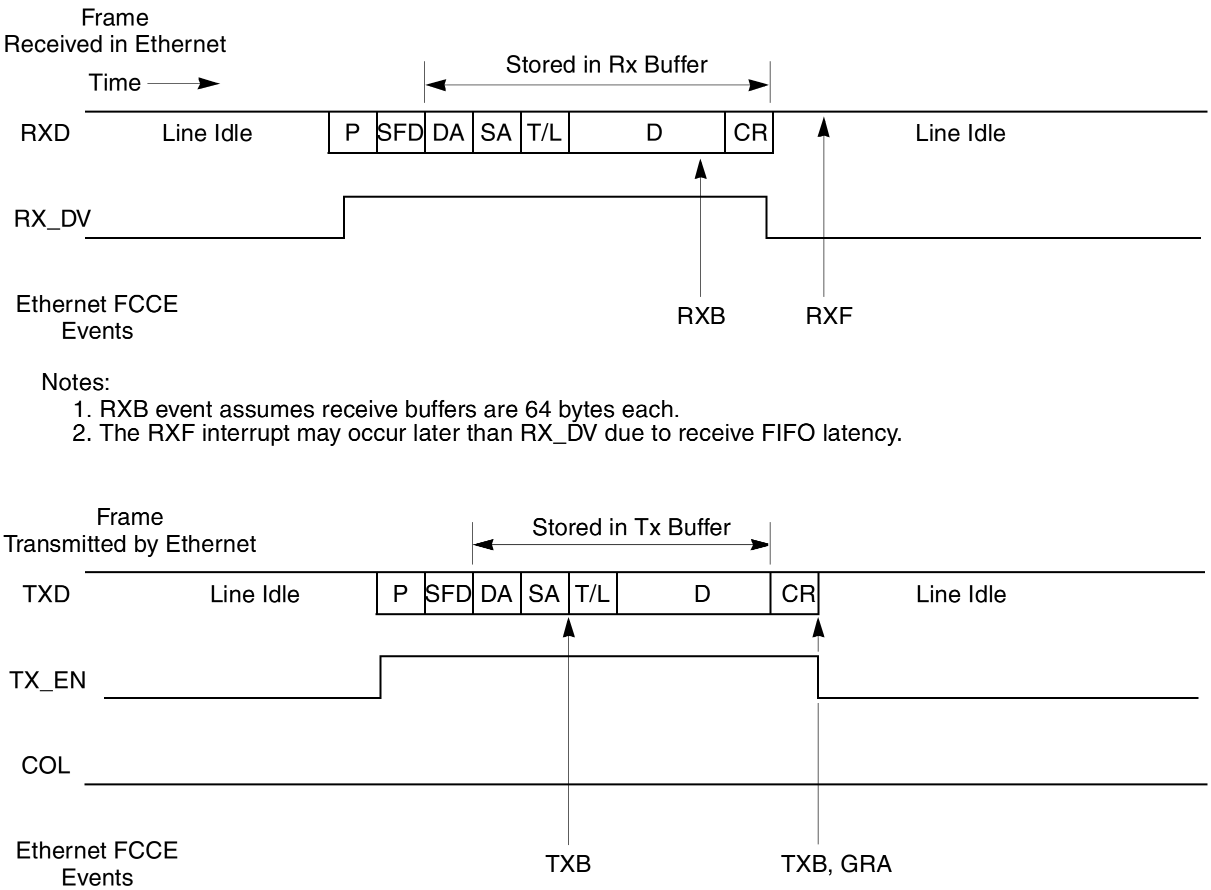
	16 31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x11312 (FCCE1), 0x11332 (FCCE2), 0x11316 (FCCM1), 0x11336 (FCCM2)

Table 32-10 describes FCCE/FCCM fields.

Table 32-10. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared
8	GRA	Graceful stop complete. A graceful stop, initiated by the GRACEFUL STOP TRANSMIT command, is complete. When the command is issued, GRA is set as soon the transmitter finishes sending a frame in progress. If no frame is in progress, GRA is set immediately.
9	RXC	RX control. A control frame has been received (FSMR[FCE] must be set). As soon as the transmitter finishes sending the current frame, a pause operation is performed.
10	TXC	TX control. An out-of-sequence frame was sent.
11	TXE	Tx error. An error occurred on the transmitter channel. This event is not maskable via the TxBD[I] bit
12	RXF	Rx frame. Set when a complete frame is received on the Ethernet channel. This event is not maskable via the RxBD[I] bit
13	BSY	Busy condition. Set when a frame is received and discarded due to a lack of buffers.
14	TXB	Tx buffer. Set when a buffer has been sent on the Ethernet channel.
15	RXB	Rx buffer. A buffer that was not a complete frame is received on the Ethernet channel.
16–31	—	Reserved, should be cleared.

Figure 32-9 shows interrupts that can be generated in the Ethernet protocol.



Notes:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RXF interrupt may occur later than RX_DV due to receive FIFO latency.

Notes:
1. TXB events assume the frame required two transmit buffers.
2. The GRA event assumes a graceful stop transmit command was issued during frame transmission.

Legend:
P = Preamble, SFD = Start Frame Delimiter, DA and SA = Destination/Source Address,
T/L = Type/Length, D = Data, CR = CRC Bytes

Figure 32-9. Ethernet Interrupt Events Example

NOTE

The FCC status register is not valid for the Ethernet protocol. The current state of the MII signals can be read through the parallel ports.

32.19 Ethernet RxBDs

The Ethernet controller uses the RxBD to report information about the received data for each buffer. Figure 32-10 shows the FCC Ethernet RxBD format.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	F	CMR	M	BC	MC	LG	NO	SH	CR	OV	CL
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 32-10. Fast Ethernet Receive Buffer (RxBD)

Table 32-11 describes Ethernet RxBD fields.

Table 32-11. RxBD Field Descriptions

Bits	Name	Description
0	**E**	Empty 0 The buffer associated with this RxBD is full or reception terminated due to an error. The core can examine or read to any fields of this RxBD. The CP does not use this BD as long as E = 0. 1 The associated buffer is empty. The RxBD and buffer are owned by the CP. Once E = 1, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in RxBD table) 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and determined only by the W bit. The RxBD table must contain more than one BD in Ethernet mode.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is used; FCCE[RXF] is unaffected. 1 FCCE[RXB] or FCCE[RXF] are set when this buffer is used by the Ethernet controller. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the Ethernet controller when this buffer is the last in a frame. This implies the end of the frame or a reception error, in which case one or more of the CL, OV, CR, SH, NO, and LG bits are set. The Ethernet controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame 1 Last buffer in a frame
5	F	First in frame. Set by the Ethernet controller when this buffer is the first in a frame. 0 Not the first buffer in a frame 1 First buffer in a frame
6	CMR	CAM match result for the frame. Set by the Ethernet controller when using a CAM for address matching and FPSMR[ECM] = 1. Valid only if the L bit is set. 0 A hit in the CAM 1 A miss in the CAM
7	M	Miss. Set by the Ethernet controller for frames that are accepted in promiscuous mode, but are flagged as a miss by the internal address recognition. Thus, while using promiscuous mode, the user uses the miss bit to determine quickly whether the frame is destined for this station. Valid only if RxBD[I] is set. 0 The frame is received because the address is recognized. 1 The frame is received because of promiscuous mode (address is not recognized).

Table 32-11. RxBD Field Descriptions (continued)

Bits	Name	Description
8	BC	Broadcast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is the broadcast address.
9	MC	Multicast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is a multicast address other than a broadcast address.
10	LG	Rx frame length violation. A frame length greater than the MFLR (maximum frame length) defined for this FCC is recognized.
11	NO	Rx nonoctet aligned frame. A frame that contained a number of bits not divisible by eight is received and the CRC check at the preceding byte boundary generated an error.
12	SH	Short frame. A frame length less than the MINFLR (minimum frame length) defined for this channel is recognized. This indication is possible only if the FPSMR[RSH] = 1.
13	CR	Rx CRC error. This frame contains a CRC error.
14	OV	Overrun. A receiver overrun occurred during frame reception.
15	CL	Collision. This frame is closed because a collision occurred during frame reception. Set only if a late collision occurs or if FPSMR[RSH] is set. The late collision definition is determined by the setting of FPSMR[LCW].

Data length is the number of octets the CP writes into this BD data buffer. It is written by the CP as the buffer is closed. When this BD is the last BD in the frame (RxBD[L] = 1), the data length contains the total number of frame octets (including four bytes for CRC). Note that at least as much memory should be allocated for each receive buffer as the size specified in MRBLR. MRBLR should be divisible by 32 and not less than 64.

The receive buffer pointer, which points to the first location of the associated data buffer, can reside in external memory. This value must be divisible by 16.

When a received frame's data length is an exact multiple of MRBLR, the last BD contains only the status and total frame length.

NOTE

At least two BDs must be prepared before beginning reception.

Figure 32-11 shows how RxBDs are used during Ethernet reception.

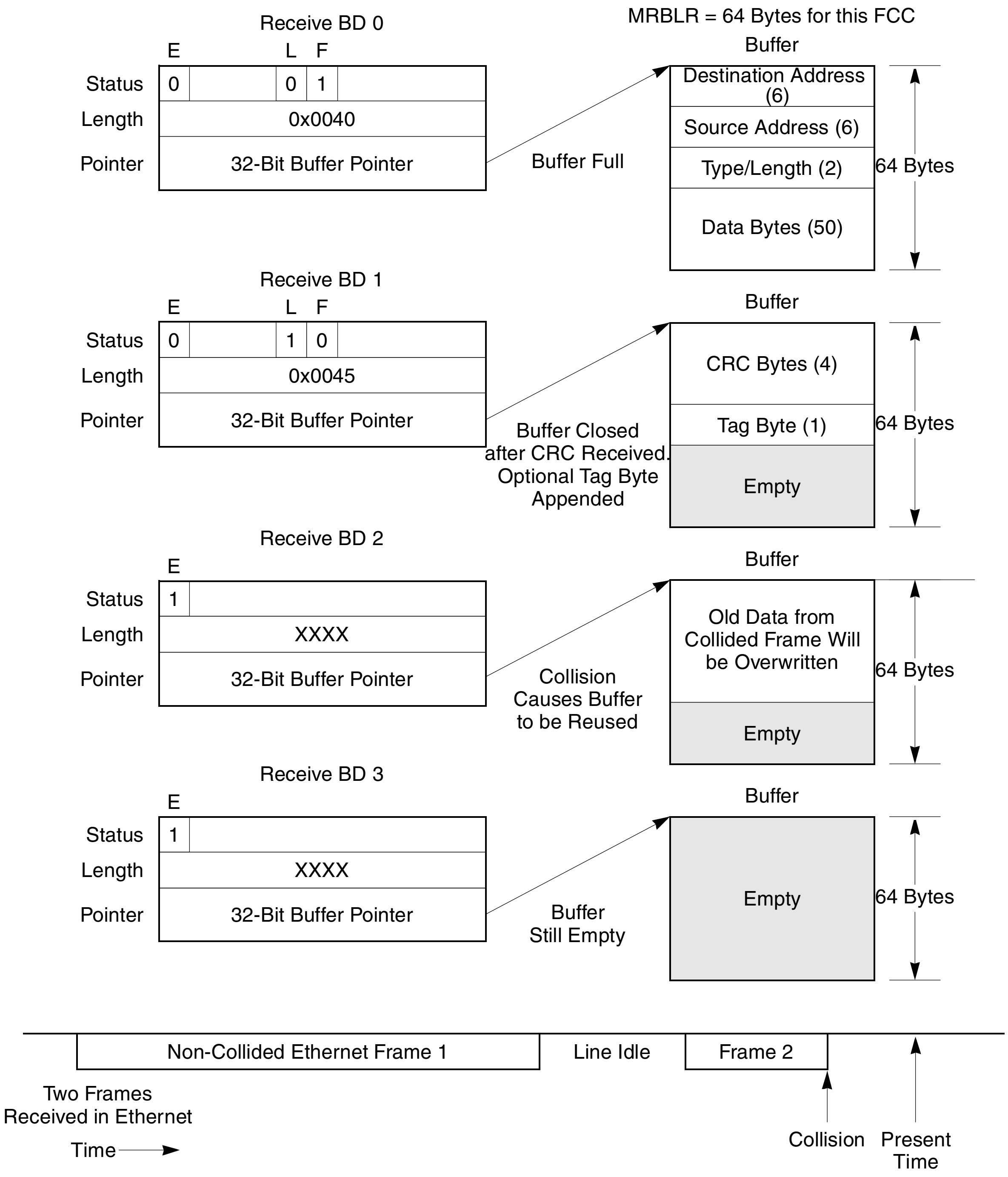


Figure 32-11. Ethernet Receiving Using RxBDs

32.20 Ethernet TxBDs

Data is sent to the Ethernet controller for transmission on an FCC channel by arranging it in buffers referenced by the channel’s TxBD table. The Ethernet controller uses TxBDs to confirm transmission or indicate errors so the core knows when buffers have been serviced. Figure 32-12 shows the FCC Ethernet TxBD format.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**R**	**PAD**	**W**	**I**	**L**	**TC**	DEF	HB	LC	RL	RC				UN	CSL
Offset + 2	**Data length**															
Offset + 4	**Tx data Buffer Pointer**															
Offset + 6																

Figure 32-12. Fast Ethernet Transmit Buffer (TxBD)

Table 32-12 describes Ethernet TxBD fields.

Table 32-12. Ethernet TxBD Field Definitions

Field	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 after an error. 1 The buffer is ready to be sent. The buffer is either waiting or in the process of being sent. The user cannot change fields in this BD or its associated buffer once R = 1.
1	**PAD**	Short frame padding. Valid only when L = 1; otherwise, it is ignored. 0 Do not add PADs to short frames. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals the MINFLR. The PAD bytes are stored in a buffer pointed to by PAD_PTR in the parameter RAM.
2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to in the table. The number of TxBDs in this table is programmable and determined only by the W bit. The TxBD table must contain more than one BD in Ethernet mode.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced; FCCE[TXE] is unaffected. 1 FCCE[TXB] or FCCE[TXE] is set after this buffer is serviced. These bits can cause interrupts if they are enabled.
4	**L**	Last 0 Not the last buffer in the transmit frame 1 Last buffer in the current transmit frame
5	**TC**	Tx CRC. Valid only when the L bit is set; otherwise, it is ignored. 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. This frame did not have a collision before it was sent but it was sent late because of deferring.
7	HB	Heartbeat. The collision input is not asserted within 40 transmit serial clocks following completion of transmission. This bit cannot be set unless FPSMR[HBC] = 1. Written by the Ethernet controller after sending the associated buffer.
8	LC	Late collision. A collision occurred after the number of bytes defined in FPSMR[LCW] (56 or 64) are sent. The Ethernet controller terminates the transmission and updates LC after sending the buffer.
9	RL	Retransmission limit. The transmitter failed (RET_LIM + 1) attempts to successfully send a message due to repeated collisions. The Ethernet controller updates RL after sending the buffer.

Table 32-12. Ethernet TxBD Field Definitions (continued)

Field	Name	Description
10–13	RC	Retry count. Indicates the number of retries required for this frame to be successfully sent. If RC = 0, the frame is sent correctly the first time. If RC = 15 and RET_LIM = 15 in the parameter RAM, 15 retries were needed. If RC = 15 and RET_LIM > 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The Ethernet controller updates UN after sending the buffer.
15	CSL	Carrier sense lost. Carrier sense is lost during frame transmission. The Ethernet controller updates CSL after sending the buffer.

Data length is the number of octets the Ethernet controller should transmit from this BD data buffer. This value should be greater than zero. The CP never modifies the data length in a TxBD.

The Tx data buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in external memory. The CP never modifies the buffer pointer.

Chapter 33
FCC HDLC Controller

Layer 2 of the seven-layer OSI model is the data link layer (DLL), in which HDLC is one of the most common protocols. The framing structure of HDLC is shown in Figure 33-1. HDLC uses a zero insertion/deletion process (commonly known as bit stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer for a method of clocking and of synchronizing the transmitter/receiver.

Because the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or a packet-and-circuit switched system, an address field is needed for the frame's destination address. The length of this field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. For instance, SDLC and LAPB use an 8-bit address and SS#7 has no address field because it is used always in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one device. It also defines a broadcast address. Some HDLC-type protocols also permit extended addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow-control number and defines the frame type (control or data). The exact use and structure of this field depends upon the protocol using the frame. Data is transmitted in the data field, which can vary in length depending upon the protocol using the frame. Layer 3 frames are carried in this data field.

Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16-bits long but can be as long as 32-bits. In HDLC, the lsb of each octet is transmitted first and the msb of the CRC is transmitted first.

When GFMR[MODE] selects HDLC mode, that FCC functions as an HDLC controller. When an FCC in HDLC mode is used with a non-multiplexed modem interface, the FCC outputs are connected directly to the external pins. Modem signals can be supported through the appropriate port pins. The receive and transmit clocks can be supplied either externally or from the bank of baud-rate generators. The HDLC controller can also be connected to one of the TDM channels of the serial interface and used with the TSA. The HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with other FCCs. The user can allocate external buffer descriptors (BDs) for receive and transmit tasks so many frames can be sent or received without core intervention.

33.1 Key Features

Key features of the HDLC include the following:

- Flexible data buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (receive and transmit)
- Received frames threshold to reduce interrupt overhead

- Four address comparison registers with masks
- Maintenance of four 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation/checking
- Detection of nonoctet-aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- External BD table
- Up to T3 rate
- Support of time stamp mode for Rx frames
- Support of nibble mode HDLC (4 bits per clocks)

33.2 HDLC Channel Frame Transmission Processing

The HDLC transmitter is designed to work with almost no core intervention. When the core enables a transmitter, it starts sending flags or idles as programmed in the HDLC mode register (FPSMR). The HDLC controller polls the first BD in the transmit channel BD table. When there is a frame to transmit, the HDLC controller fetches the data (address, control, and information) from the first buffer and begins sending the frame after first inserting the user-specified minimum number of flags between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the FCC appends the CRC (if selected) and closing flag. In HDLC, the lsb of each octet and the msb of the CRC are sent first. Figure 33-1 shows a typical HDLC frame.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 bits	16 bits	8 bits	8*n* bits	16 bits	8 bits

Figure 33-1. HDLC Framing Structure

After the closing flag is sent, the HDLC controller writes the frame status bits into the BD and clears the R bit. When the end of the current BD is reached and the L (last) bit is not set (working in multibuffer mode), only the R bit is cleared. In either mode, an interrupt can be issued if the I bit in the TxBD is set. The HDLC controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each buffer, after a specific buffer, after each frame, or after a number of frames.

To rearrange the transmit queue before the CP has sent all buffers, issue the STOP TRANSMIT command. This can be useful for sending expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller aborts the current frame transmission and starts transmitting idles or flags. When the HDLC controller is given the RESTART TRANSMIT command, it resumes transmission. To insert a high-priority frame without aborting the current frame, the GRACEFUL STOP TRANSMIT command can be issued. A special interrupt (GRA) can be generated in the event register when the current frame is complete.

33.3 HDLC Channel Frame Reception Processing

The HDLC receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, and maximum frame length checking. The received frame is available for any HDLC-based protocol. When the core enables a receiver, the receiver waits for an opening flag character. When it detects the first byte of the frame, the HDLC controller compares the frame address against the user-programmable addresses. The user has four 16-bit address registers and an address mask available for address matching. The HDLC controller compares the received address field to the user-defined values after masking with the address mask. The HDLC controller can also detect broadcast (all ones) address frames if one address register is written with all ones.

If a match is detected, the HDLC controller checks the prefetched BD; if it is empty, it starts transferring the incoming frame to the BD's associated buffer. When the buffer is full, the HDLC controller clears BD[E] and generates an interrupt if BD[I] = 1. If the incoming frame is larger than the buffer, the HDLC controller fetches the next BD in the table and, if it is empty, continues transferring the frame to the associated buffer.

During this process, the HDLC controller checks for frames that are too long. When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. The data length written to the last BD in the HDLC frame is the length of the entire frame. This enables HDLC protocols that lose frames to correctly recognize a frame-too-long condition.

The HDLC controller then sets the last buffer in frame bit, writes the frame status bits into the BD, and clears the E bit and fetches the next BD. The HDLC controller then generates a maskable interrupt, indicating that a frame was received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames can be received separated only by a single shared flag.

The user can configure the HDLC controller not to interrupt the core until a specified number of frames have been received. This is configured in the received frames threshold (RFTHR) location of the parameter RAM. This function can be combined with a timer to implement a time-out if fewer than the threshold number of frames are received.

33.4 HDLC Parameter RAM

When an FCC operates in HDLC mode, the protocol-specific area of the FCC parameter RAM is mapped with the HDLC-specific parameters in Table 33-1.

Table 33-1. FCC HDLC-Specific Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x3C	—	2 words	Reserved
0x44	**C_MASK**	Word	CRC constant. For the 16-bit CRC-CCITT, initialize C_MASK to 0x0000_F0B8. For the 32-bit CRC-CCITT, initialize C_MASK to 0xDEBB_20E3.
0x48	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize C_PRES to 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize C_PRES to 0xFFFF_FFFF.
0x4C	**DISFC**[2]	Hword	Discard frame counter. Counts error-free frames discarded due to lack of buffers.

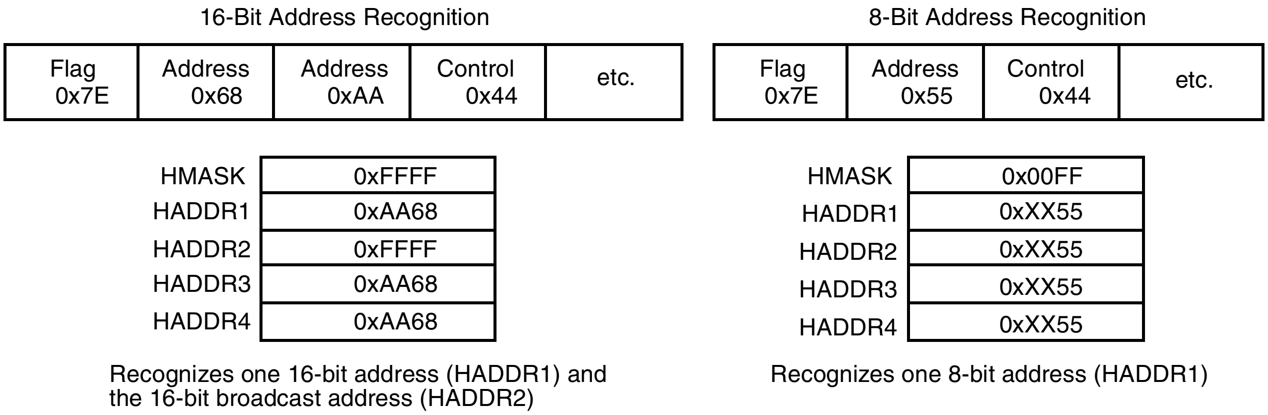
Table 33-1. FCC HDLC-Specific Parameter RAM Memory Map (continued)

Offset[1]	Name	Width	Description
0x4E	**CRCEC**[2]	Hword	CRC error counter. Counts frames not addressed to the user or frames received in the BSY condition, but does not include overrun, $\overline{\text{CD}}$ lost, or abort errors.
0x50	**ABTSC**[2]	Hword	Abort sequence counter
0x52	**NMARC**[2]	Hword	Nonmatching address Rx counter. Counts nonmatching addresses received (error-free frames only). See the HMASK and HADDR[1–4] parameter description.
0x54	MAX_CNT	Word	Max_length counter. Temporary decrementing counter that tracks frame length.
0x58	**MFLR**	Hword	Max frame length register. If the HDLC controller detects an incoming HDLC frame that exceeds the user-defined value in MFLR, the rest of the frame is discarded and the LG (Rx frame too long) bit is set in the last BD belonging to that frame. The HDLC controller waits for the end of the frame and then reports the frame status and length in the last RxBD. MFLR includes all in-frame bytes between the opening and closing flags (address, control, data, and CRC).
0x5A	**RFTHR**	Hword	Received frames threshold. Used to reduce the interrupt overhead that might otherwise occur when a series of short HDLC frames arrives, each causing an RXF interrupt. By programming RFTHR, the user lowers the frequency of RXF interrupts, which occur only when the RFTHR value is reached. Note that the user should provide enough empty RxBDs to receive the number of frames specified in RFTHR.
0x5C	**RFCNT**	Hword	Received frames count. A decrementing counter used to implement this feature. Initialize this counter with RFTHR.
0x5E	**HMASK**	Hword	HMASK and HADDR[1–4]. The HDLC controller reads the frame address from the HDLC receiver, checks it against the four address register values, and masks the result with HMASK. In HMASK, a 1 represents a bit position for which address comparison should occur; 0 represents a masked bit position. When addresses match, the address and subsequent data are written into the buffers. When addresses do not match and the frame is error-free, the nonmatching address received counter (NMARC) is incremented. Note that for 8-bit addresses, mask out (clear) the eight high-order bits in HMASK. The eight low-order bits and HADDRx should contain the address byte that immediately follows the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDRx should contain 0xAA68 and HMASK should contain 0xFFFF. See Figure 33-2.
0x60	**HADDR1**	Hword	
0x62	**HADDR2**	Hword	
0x64	**HADDR3**	Hword	
0x66	**HADDR4**	Hword	
0x68	TS_TMP	Hword	Temporary storage
0x6A	TMP_MB	Hword	Temporary storage

[1] Offset from FCC base: 0x8400 (FCC1) and 0x8500 (FCC2); see Section 13.5.2, “Parameter RAM.”

[2] DISFC, CRCEC, ABTSC, and NMARC—These 16-bit (modulo 216) counters are maintained by the CP. The user can initialize them while the channel is disabled.

Figure 33-2 shows an example of using HMASK and HADDR[1–4].



16-Bit Address Recognition

Flag 0x7E	Address 0x68	Address 0xAA	Control 0x44	etc.

HMASK	0xFFFF
HADDR1	0xAA68
HADDR2	0xFFFF
HADDR3	0xAA68
HADDR4	0xAA68

Recognizes one 16-bit address (HADDR1) and the 16-bit broadcast address (HADDR2)

8-Bit Address Recognition

Flag 0x7E	Address 0x55	Control 0x44	etc.

HMASK	0x00FF
HADDR1	0xXX55
HADDR2	0xXX55
HADDR3	0xXX55
HADDR4	0xXX55

Recognizes one 8-bit address (HADDR1)

Figure 33-2. HDLC Address Recognition Example

33.5 Programming Model

The core configures each FCC to operate in the protocol specified in GFMR[MODE]. The HDLC controller uses the same data structure as other modes. This data structure supports multibuffer operation and address comparisons.

33.5.1 HDLC Command Set

The transmit and receive commands are issued to the CPCR; see Section 13.4, “Command Set.”

Table 33-2 describes the transmit commands that apply to the HDLC controller.

Table 33-2. Transmit Commands

Command	Description
STOP TRANSMIT	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in transmit enable mode and starts polling the first BD in the table every 256 transmit clocks (immediately if FTODR[TOD] = 1). STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the HDLC controller during frame transmission, transmission is aborted after a maximum of 64 additional bits are sent and the transmit FIFO buffer is flushed. The TBPTR is not advanced, no new BD is accessed, and no new frames are sent for this channel. The transmitter sends an abort sequence consisting of 0x7F (if the command was given during frame transmission) and begins sending flags or idles, as indicated by the HDLC mode register. Note that if FPSMR[MFF] = 1, one or more small frames can be flushed from the transmit FIFO buffer. The GRACEFUL STOP TRANSMIT command can be used to avoid this.
GRACEFUL STOP TRANSMIT	Used to stop transmission smoothly rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame finishes sending or immediately if no frame is being sent. FCCE[GRA] is set once transmission has stopped. Then the HDLC transmit parameters (including BDs) can be modified. The TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and the RESTART TRANSMIT command is issued.

Table 33-2. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Enables character transmission on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command is issued and the channel in its FCC mode register is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or $\overline{\text{CTS}}$ lost with no automatic frame retransmission). The HDLC controller resumes sending from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in this serial channel parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Notice that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Table 33-3 describes the receive commands that apply to the HDLC controller.

Table 33-3. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in receive enable mode and uses the first BD in the table. The ENTER HUNT MODE command is generally used to force the HDLC receiver to abort reception of the current frame and enter the hunt mode. In hunt mode, the HDLC controller continually scans the input data stream for the flag sequence. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should be issued only when the receiver is disabled. Notice that the INIT TX AND RX PARAMETERS command resets both receive and transmit parameters.

33.5.2 HDLC Error Handling

The HDLC controller reports frame reception and transmission error conditions using the channel BDs, error counters, and HDLC event register (FCCE). Table 33-4 describes HDLC transmission errors, which are reported through the TxBD.

Table 33-4. HDLC Transmission Errors

Error	Description
Transmitter underrun	When this error occurs, the channel terminates buffer transmission, transmit an ABORT sequence (a sequence that will generate CRC error on the frame), closes the buffer, sets the underrun (U) bit in the BD, and generates the TXE interrupt if it is enabled. The channel resumes transmission after receiving the RESTART TRANSMIT command.
$\overline{\text{CTS}}$ lost during frame transmission	When this error occurs, the channel terminates buffer transmission, closes the buffer, sets TxBD[CT], and generates a TXE interrupt (if it is enabled). The channel resumes transmission after receiving the RESTART TRANSMIT command.

Table 33-5 describes HDLC reception errors, which are reported through the RxBD.

Table 33-5. HDLC Reception Errors

<table>
<tr><th>Error</th><th>Description</th></tr>
<tr><td>Overrun error</td><td>The HDLC controller maintains an internal FIFO buffer for receiving data. The CP begins programming the SDMA channel and updating the CRC whenever data is received in the FIFO buffer. When a receive FIFO overrun occurs, the channel writes the received data byte to the internal FIFO buffer over the previously received byte. The previous byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates the RXF interrupt if it is enabled. The receiver then enters hunt mode. Even if the overrun occurs during a frame whose address is not matched in the address recognition logic, an RxBD with data length two is opened to report the overrun and the RXF interrupt is generated if it is enabled.</td></tr>
<tr><td>$\overline{CD}$ lost during frame reception</td><td>When this error occurs, the channel terminates frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if it is enabled. This error has highest priority. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode. If $\overline{CD}$ is Lost during the first 8 serial bits it will not be reported as $\overline{CD}$ Lost error and there will be no indication of error.</td></tr>
<tr><td>Abort sequence</td><td>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.</td></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>

33.6 HDLC Mode Register (FPSMR)

When an FCC is configured for HDLC mode, the FPSMR is used as the HDLC mode register, shown in Figure 33-3.

	0–3	4	5	6–8	9	10–15
Field	NOF	FSE	MFF	—	TS	—
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x11304 (FPSMR1), 0x11324 (FPSMR2), 0x11324 (FPSMR3)					

	16	17–23	24–25	26–31
Field	NBL	—	CRC	—
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x11306 (FPSMR1), 0x11326 (FPSMR2), 0x11326 (FPSMR3)			

Figure 33-3. HDLC Mode Register (FPSMR)

The FPSMR fields are described in Table 33-6.

Table 33-6. FPSMR Field Descriptions [1]

Bits	Name	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames (0–15 flags). If NOF = 0000, no flags are inserted between the frames. Thus, for back-to-back frames, the closing flag of one frame is immediately followed by the opening flag of the next frame.
4	FSE	Flag sharing enable. This bit is valid only if GFMR[RTSM] is set. 0 Normal operation 1 If NOF = 0000, a single shared flag is transmitted between back-to-back frames. Other values of NOF are decremented by 1 when FSE is set. This is useful in signaling system #7 applications.
5	MFF	Multiple frames in FIFO. Setting MFF applies only when in $\overline{\text{RTS}}$ mode (GFMR*x*[RTSM] = 1). 0 Normal operation. The transmit FIFO buffer must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. The receiver is not affected by this bit. 1 The transmit FIFO buffer can contain multiple frames, but lost $\overline{\text{CTS}}$ is not guaranteed to be reported on the exact buffer/frame it occurred on. This option, however, can improve the performance of HDLC transmissions for small back-to-back frames or if the user prefers to strongly limit the number of flags sent between frames. MFF does not affect the receiver. Refer to note 1 at the end of this table.
7–8	—	Reserved, should be cleared
9	TS	Time stamp 0 Normal operation 1 A 32-bit time stamp is added at the beginning of the receive BD data buffer, thus the buffer pointer must be (32-byte aligned – 4). The BD's data length does not include the time stamp. See Section 13.3.7, "RISC Time-Stamp Control Register (RTSCR)."
10–15	—	Reserved, should be cleared

Table 33-6. FPSMR Field Descriptions (continued)[1]

Bits	Name	Description
16	NBL	Nibble mode enable 0 Nibble mode disabled (1 bit of data per clock). Note that at the end of the frame (after the closing flag), $\overline{\text{RTS}}$ negates immediately after the active edge of TCLK. 1 Nibble mode enabled (4 bits of data per clock). The negation of the $\overline{\text{RTS}}$ output signal is not synchronized to the serial clock. The $\overline{\text{RTS}}$ is negated after the last nibble of the data and always before the next edge of the serial clock. Note that at the end of the frame (after the closing flag), $\overline{\text{RTS}}$ negates a maximum of 5 CPM clocks after the active edge of TCLK.
17–23	—	Reserved, should be cleared
24-25	CRC	CRC selection 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1 01 Reserved 10 32-bit CCITT-CRC (Ethernet and HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1 11 Reserved
26–31	—	Reserved, should be cleared

[1] When operating an FCC in HDLC nibble mode with the multiframe per FIFO bit off (FPSMR[MFF] = 0), the CPM might lose synchronization with the FCC HDLC controller. As a result the HDLC controller will become stuck and stop transmission. Therefore in HDLC nibble mode, FPSMR[MFF] must be set or the FCC must alternatively operate in HDLC bit mode.

33.7 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 33-4 shows an example of the RxBD process.

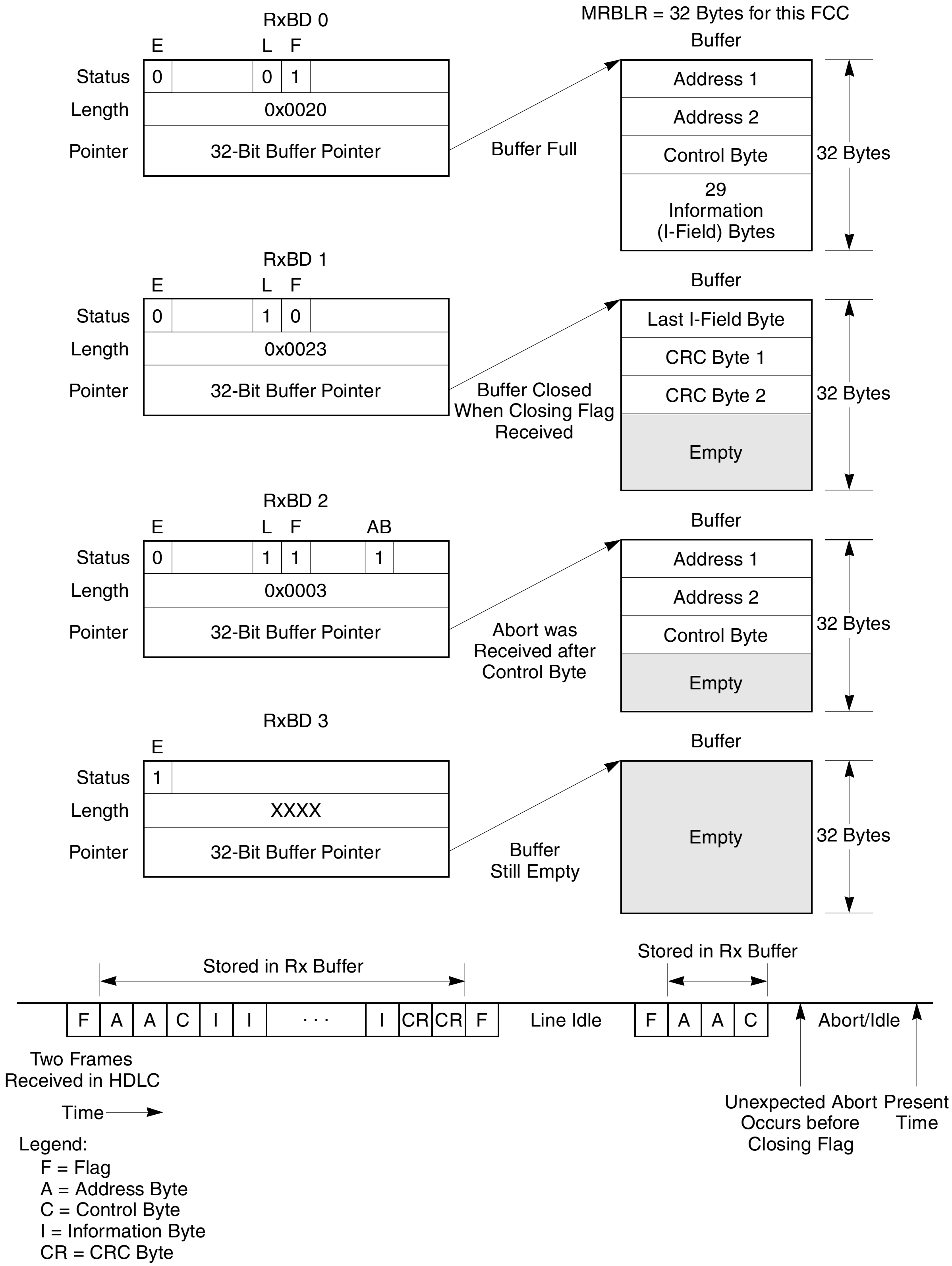


Figure 33-4. FCC HDLC Receiving Using RxBDs

Figure 33-5 shows the FCC HDLC 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**	—			LG	NO	AB	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Data Buffer Pointer**															
Offset + 6																

Figure 33-5. FCC HDLC Receive Buffer Descriptor (RxBD)

Table 33-7 describes RxBD fields.

Table 33-7. 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 CP does not use this BD while E = 0. 1 The buffer associated with this BD is empty. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in table) 0 Not the last BD in the RxBD table 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and is determined only by the W bit and the overall space constraints of the dual-port RAM. The RxBD table must contain more than one BD in HDLC mode.
3	**I**	Interrupt 0 The RXB bit is not set after this buffer is used, but RXF operation remains unaffected. 1 FCCE[RXB] or FCCE[RXF] is set when the HDLC controller uses this buffer. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the HDLC controller when this buffer is the last one in a frame. This implies the reception of a closing flag or reception of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame 1 Last buffer in a frame
5	F	First in frame. Set by the HDLC controller when this buffer is the first in a frame. 0 Not the first buffer in a frame 1 First buffer in a frame
6	**CM**	Continuous mode 0 Normal operation 1 The E bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be automatically overwritten the next time the CP accesses this BD. However, the E bit is cleared if an error occurs during reception, regardless of the CM bit.
7–9	—	Reserved, should be cleared
10	LG	Rx frame length violation. A frame length greater than the maximum defined for this channel is recognized, and only the maximum-allowed number of bytes (MFLR) is written to the data buffer. This event is not reported until the RxBD is closed, the RXF bit is set, and the closing flag is received. The number of bytes received between flags is written to the data length field of this BD.

Table 33-7. 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. $\overline{\text{CD}}$ 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 CP writes into this BD’s data buffer. It is written by the CP once the BD is closed. When this is the last BD in the frame (L = 1), this field contains the total number of frame octets, including 2 or 4 bytes for CRC. The memory allocated for this buffer should be no smaller than the MRBLR value.
- Rx data buffer pointer. The receive buffer pointer, which always points to the first location of the associated data buffer, resides in internal or external memory and must be divisible by 32 unless FPSMR[TS] = 1 (see Table 33-6).

33.8 HDLC Transmit Buffer Descriptor (TxBD)

Data is presented to the HDLC controller for transmission on an FCC channel by arranging it in buffers referenced by the channel TxBD table. The HDLC controller confirms transmission (or indicates errors) using the BDs to inform the core that the buffers have been serviced. Figure 33-6 shows the FCC HDLC TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	—							UN	CT
Offset + 2	**Data Length**															
Offset + 4	**Tx Data Buffer Pointer**															
Offset + 6																

Figure 33-6. FCC HDLC Transmit Buffer Descriptor (TxBD)

Table 33-8 describes HDLC TxBD fields.

Table 33-8. 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 CP clears R after the buffer has been sent or an error occurs. 1 The buffer is ready to be sent. The transmission may have begun, but it has not completed. The user cannot set fields in this BD once R is set.
1	—	Reserved, should be cleared
2	**W**	Wrap (final BD in table) 0 Not the last BD in the TxBD table 1 Last BD in the TxBD table. After this buffer has been used, the CP sends data from the first BD that TBASE points to in the table. The number of TxBDs in this table is determined only by the W bit and the overall space constraints of the dual-port RAM.
3	**I**	Interrupt 0 No interrupt is generated after this buffer is serviced; FCCE[TXE] is unaffected. 1 Either FCCE[TXB] or FCCE[TXE] is set when this buffer is serviced by the HDLC controller. These bits can cause interrupts if they are enabled.
4	**L**	Last 0 Not the last buffer in the frame 1 Last buffer in the current frame
5	**TC**	Tx CRC.Valid only when the L bit is set. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode 0 Normal operation 1 The R bit is not cleared by the CP after this BD is closed, allowing the buffer to be retransmitted automatically the next time the CP accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit.
7–13	—	Reserved, should be cleared
14	UN	Underrun. The HDLC controller encounters a transmitter underrun condition while sending the buffer. The HDLC controller writes UN after sending the buffer.
15	CT	$\overline{\text{CTS}}$ lost. Set when $\overline{\text{CTS}}$ is lost during frame transmission in NMSI mode. If data from more than one buffer is in the FIFO buffer when this error occurs, CT is set in the currently open TxBD. The HDLC controller writes CT after sending the buffer.

The TxBD status bits are written by the HDLC controller after sending the associated data buffer.

The remaining TxBD parameters are as follows:

- Data length is the number of bytes the HDLC controller should transmit from this data buffer; it is never modified by the CP. The value of this field should be greater than zero.
- Tx data buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in internal or external memory. This value is never modified by the CP.

33.9 HDLC Event Register (FCCE)/Mask Register (FCCM)

The FCCE is used as the HDLC event register when the FCC operates as an HDLC controller. The FCCE reports events recognized by the HDLC channel and generates interrupts. On recognition of an event, the HDLC controller sets the corresponding FCCE bit. FCCE bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the CP clears the internal interrupt request.

Interrupts generated by the FCCE can be masked in the HDLC mask register (FCCM), which has the same bit format as FCCE. If an FCCM bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked.

Figure 33-7 represents the FCC/FCCM.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—								GRA	—		TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11310 (FCCE1), 0x11330 (FCCE2), 0x11314 (FCCM1), 0x11334 (FCCM2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—						FLG	IDL	—							
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x11312 (FCCE1), 0x11332 (FCCE2), 0x11316 (FCCM1), 0x11336 (FCCM2)															

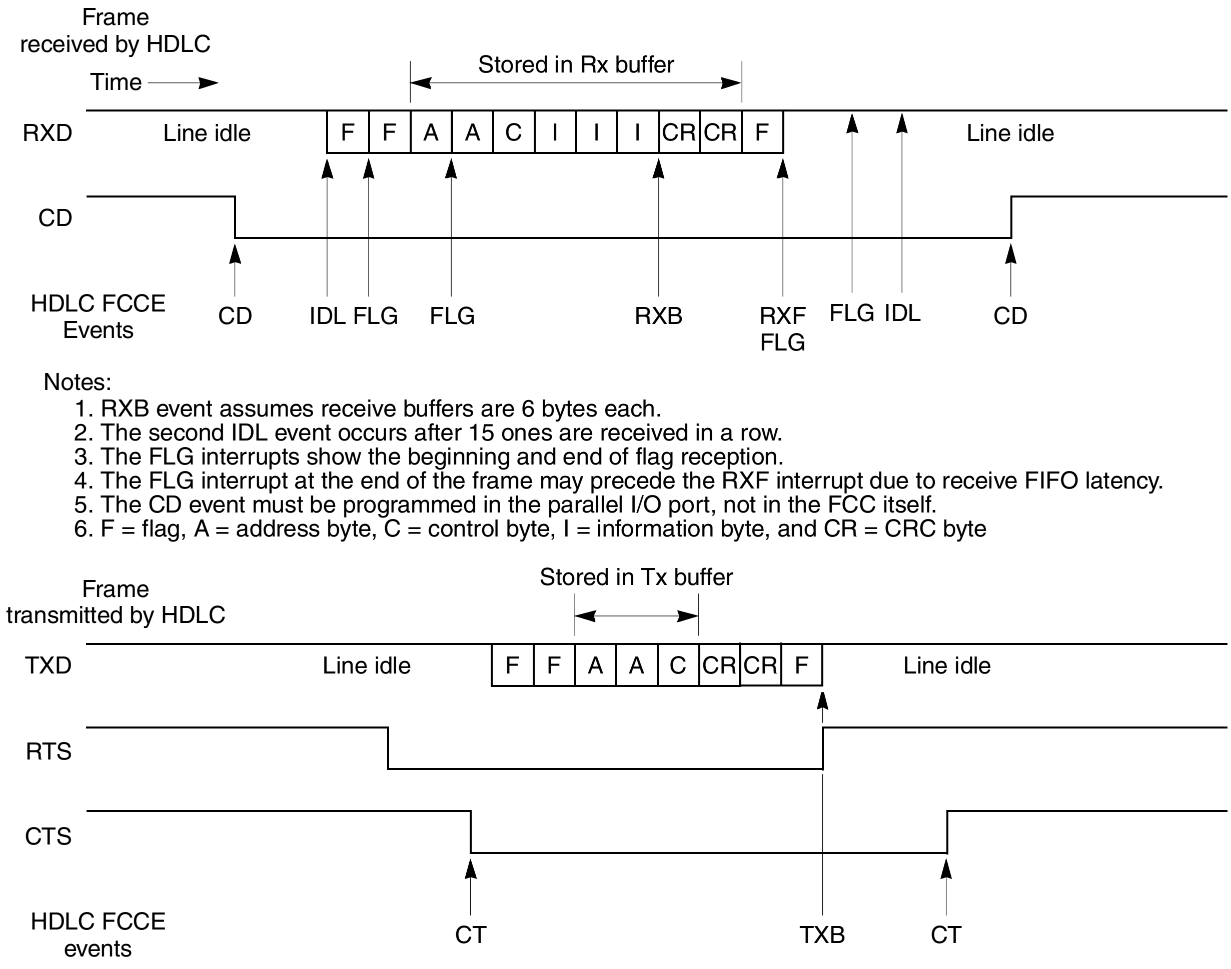
Figure 33-7. HDLC Event Register (FCCE)/Mask Register (FCCM)

Table 33-9 describes FCCE/FCCM fields.

Table 33-9. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared
8	GRA	Graceful stop complete. A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. GRA is set as soon as the transmitter finishes transmitting any frame that is in progress when the command was issued. It is set immediately if no frame is in progress when the command is issued.
9–10	—	Reserved, should be cleared
11	TXE	Tx error. An error ($\overline{\text{CTS}}$ lost or underrun) occurs on the transmitter channel. This event is not maskable via the TxBD[I] bit.
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. When RXB = 1, a buffer for which the I bit is set in the corresponding BD was filled, regardless if the end of a frame was completed in it.
16–21	—	Reserved, should be cleared
22	FLG	Flag status changed. The HDLC controller stops or starts receiving HDLC flags. The real-time status can be obtained in FCCS; see Section 33.10, “FCC Status Register (FCCS).”
23	IDL	Idle sequence status changed. A change in the status of the serial line is detected on the HDLC line. The real-time status can be read in FCCS; see Section 33.10, “FCC Status Register (FCCS).”
24–31	—	Reserved, should be cleared

Figure 33-8 shows interrupts that can be generated in the HDLC protocol.



Notes:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.
3. The FLG interrupts show the beginning and end of flag reception.
4. The FLG interrupt at the end of the frame may precede the RXF interrupt due to receive FIFO latency.
5. The CD event must be programmed in the parallel I/O port, not in the FCC itself.
6. F = flag, A = address byte, C = control byte, I = information byte, and CR = CRC byte

Notes:
1. TXB event shown assumes all three bytes were put into a single buffer.
2. Example shows one additional opening flag. This is programmable.
3. The CT event must be programmed in the parallel I/O port, not in the FCC itself.

Figure 33-8. HDLC Interrupt Event Example

33.10 FCC Status Register (FCCS)

The FCCS register, shown in Figure 33-9, allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ signals are part of the parallel I/O port; see Chapter 37, “Parallel I/O Ports.”

	0–4	5	6	7
Field	—	FG	—	ID
Reset	0000_0000			
R/W	R			
Addr	0x11318 (FCCS1), 0x11338 (FCCS2)			

Figure 33-9. FCC Status Register (FCCS)

Table 33-10 describes FCCS bits.

Table 33-10. FCCS Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. While FG is cleared, each time a new bit is received the most recently received 8 bits are examined to see if a flag is present. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once FG is set, it remains set at least 8 bit times while the next 8 bits of input data are examined. If another flag occurs, FG stays set for at least another 8 bits. Otherwise, FG is cleared and the search begins again. 0 HDLC flags are not currently being received. 1 HDLC flags are currently being received.
6	—	Reserved, should be cleared.
7	ID	Idle status. ID is set when the RXD signal is a logic one for 15 or more consecutive bit times; it is cleared after a logic zero is received. 0 The line is busy. 1 The line is idle.

Chapter 34
FCC Transparent Controller

The FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter. Transparent mode provides a clear channel on which the FCC performs no bit-level manipulation—implementing higher-level protocols would require software. Transparent mode is also referred to as a totally transparent or promiscuous operation.

Basic applications for an FCC in transparent mode include the following:

- For data, such as voice, moving serially without the need for protocol processing
- For board-level applications, such as chip-to-chip communications, requiring a serial-to-parallel and parallel-to-serial conversion
- For applications requiring the switching of data paths without altering the protocol encoding itself, such as a multiplexer in which data from a high-speed TDM serial stream is divided into multiple low-speed data streams

An FCC transmitter and receiver can be programmed in transparent mode independently. Setting GFMR*x*[TTx] enables the transparent transmitter; setting GFMR*x*[TRx] enables the transparent receiver. Both bits must be set for full-duplex transparent operation. If only one bit is set, the other half of the FCC operates with the protocol programmed in GFMR*x*[MODE]. This allows loopback modes to transfer data from one memory location to another (using DMA) while the data is converted to a specific serial format. However, the Ethernet and ATM controllers cannot be split in this way. See Section 29.2, “Mode Registers.”

The FCC in transparent mode can work with the TSA or NMSI and support modem lines using the general-purpose I/O signals. The data can be transmitted and received with msb or lsb first in each octet. The FCC consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to the other FCCs. Each clock can be supplied from the internal BRG bank or external signals.

34.1 Features

The following is a list of the transparent controller’s important features:

- Flexible data buffers
- Automatic SYNC detection on receive
  - 16-bit pattern
  - 8-bit pattern
  - Automatic sync (always synchronized)
  - External sync signal support
- CRCs can optionally be transmitted and received

- Reverse data mode
- Another protocol can be performed on the FCC’s other half (transmitter or receiver) during transparent mode
- External BD table

34.2 Transparent Channel Operation

The transparent transmitter and receiver operates in the same way as the HDLC controller of the FCC (see Chapter 33, “FCC HDLC Controller,”) except in the following ways:

1. The FPSMR does not affect the transparent controller, only the GFMR does.
2. In Table 33-1, MFLR, HMASK, RFTHR, and RFCNT must be cleared for proper operation of the transparent receiver.
3. Transmitter synchronization has to be achieved using $\overline{\text{CTS}}$ before the transmitter begins sending; see Section 34.3, “Achieving Synchronization in Transparent Mode.”

34.3 Achieving Synchronization in Transparent Mode

Once the FCC transmitter is enabled for transparent operation in the GFMR, the TxBD is prepared for the FCC, and the transmit FIFO is preloaded by the SDMA channel, transmit synchronization must be established before data can be sent.

Similarly, once the FCC receiver is enabled for transparent operation in the GFMR and the RxBD is made empty for the FCC, receive synchronization must occur before data can be received. The synchronization process gives the user bit-level control of when the transmission and reception begins. The methods for this are as follows:

- An in-line synchronization pattern
- External synchronization signals
- Automatic sync

34.3.1 In-Line Synchronization Pattern

The transparent channel can be programmed to transmit and receive a synchronization pattern if GFMR[SYNL] ≠ 0; see Section 29.2, “Mode Registers.” The pattern is defined in the FDSR; see Section 29.4, “FCC Data Synchronization Registers (FDSRx).” GFMR[SYNL] defines the SYNC pattern length. The synchronization pattern is shown in Figure 34-1.

	0 … 7	8 … 15
Field	8-Bit Sync Pattern	—
Field	16-Bit Sync Pattern (Second Byte)	 (First Byte)

Figure 34-1. In-Line Synchronization Pattern

The receiver synchronizes on the synchronization pattern located in the FDSR. For instance, if an 8-bit SYNC is selected, reception begins as soon as these eight bits are received, beginning with the first bit following the 8-bit SYNC. This effectively links the transmitter synchronization to the receiver synchronization.

34.3.2 External Synchronization Signals

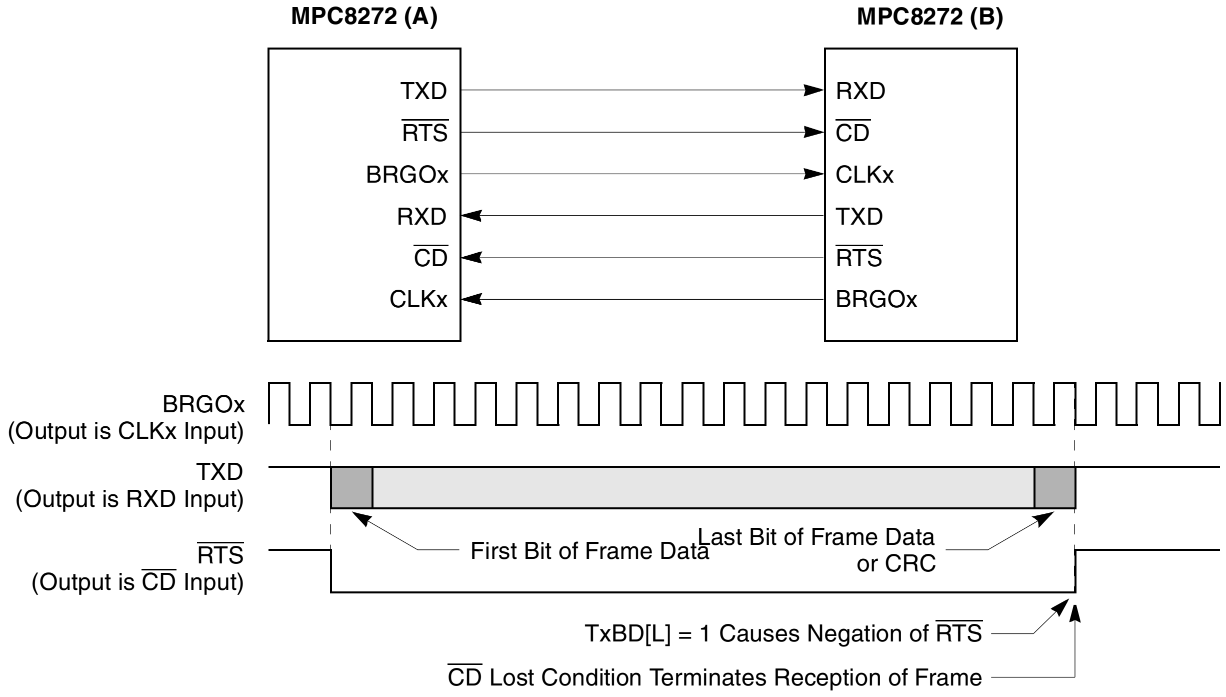
If GFMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options.

- The pulse/envelope option determines whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must be asserted and stay that way for the duration of the transparent frame. This option is controlled by the CDP and CTSP bits of the GFMR. If the user expects a continuous stream of data without interruption, the pulse option should be used. However, if the user needs to identify frames of transparent data, the envelope mode of the these signals should be used. Note that the first bit of a frame is transmitted as zero every time $\overline{\text{RTS}}$ is asserted before $\overline{\text{CTS}}$ is asserted (GFMR[CTSS] = 1); subsequent data bits are sent accurately. Similarly, if $\overline{\text{CTS}}$ is in pulse mode (GFMR[CTSP] = 1), only the first frame is affected. If $\overline{\text{CTS}}$ is not in pulse mode (GFMR[CTSP] = 0), every frame is affected separately. Note that if NRZI encoding is used (GFMR[TENC]=01), $\overline{\text{RTS}}$ must be asserted before $\overline{\text{CTS}}$, or else the first bit of the frame might be corrupted.
- The sampling option determines the delay between $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ being asserted and the resulting action by the FCC. These signals can be assumed to be asynchronous to the data and then internally synchronized by the FCC, or they can be assumed to be synchronous to the data giving faster operation. This option allows the $\overline{\text{RTS}}$ of one FCC to be connected to the $\overline{\text{CD}}$ of another FCC (on another MPC8272) and to have the data synchronized and bit aligned. It is also an option to link the transmitter synchronization to the receiver synchronization.

When working with the FCC receiver in envelope mode, $\overline{\text{RTS}}$ should be asserted for at least 3 clock cycles between frames. Otherwise, the receiver cannot recognize the start of a new frame. Diagrams for the pulse/envelope and sampling options are in Section 29.11, “FCC Timing Control.”

34.3.3 Transparent Synchronization Example

Figure 34-2 shows an example of synchronization using external signals.



Notes:

[1] Each MPC8272 generates its own transmit clocks. If the transmit and receive clocks are the same, one can generate transmit and receive clocks for the other MPC8272. For example, CLKx on MPC8272 (B) could be used to clock the transmitter and receiver

[2] $\overline{\text{CTS}}$ should be configured as always asserted in the parallel I/O port or connected to ground externally.

[3] The required GSMR configurations are DIAG= 00, CTSS=1, CTSP is a do not care, CDS=1, CDP=0, TTX=1, and TRX=1. REVD and TCRC are application-dependent.

[4] The transparent frame contains a CRC if TxBD[TC] is set.

Figure 34-2. Sending Transparent Frames between MPC8272s

MPC8272(A) and MPC8272(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 MPC8272’s $\overline{\text{CD}}$. GFMR[SYNL] is not used and transmission and reception from each MPC8272 are independent.

Chapter 35
Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the MPC8272 to exchange data between other MPC8272 chips, the MPC860, the MC68360, the MC68302, the M68HC11 and M68HC05 microcontroller families, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock and slave select). The SPI block consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously.

Because the SPI receiver and transmitter are double-buffered, as shown in Figure 35-1, the effective FIFO size (latency) is 2 characters. The SPI's msb is shifted out first. When the SPI is disabled in the SPI mode register (SPMODE[EN] = 0), it consumes little power.

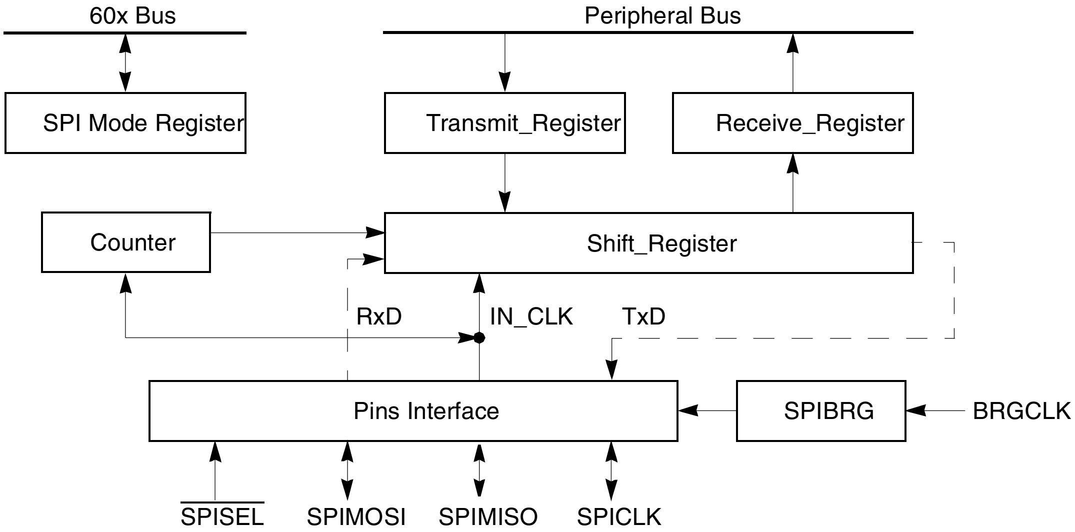


Figure 35-1. SPI Block Diagram

35.1 Features

The following is a list of the SPI's main features:

- Four-signal interface (SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$) multiplexed with port D signals
- Full-duplex operation
- Works with data characters from 4 to 16 bits long

- Supports back-to-back character transmission and reception
- Master or slave SPI modes supported
- Multiple master environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports maximum clock rates of 25 MHz in master mode and 50 MHz in slave mode, assuming a 100-MHz system clock
- Independent programmable baud rate generator
- Programmable clock phase and polarity
- Open-drain outputs support multiple master configuration
- Local loopback capability for testing

35.2 SPI Clocking and Signal Functions

The SPI can be configured as a slave or as a master in single- or multiple-master environments. The master SPI generates the transfer clock SPICLK using the SPI baud rate generator (BRG). The SPI BRG takes its input from BRGCLK, which is generated in the MPC8272 clock synthesizer.

SPICLK is a gated clock, active only during data transfers. Four combinations of SPICLK phase and polarity can be configured with SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multiple-master configuration in which a shared SPI signal is driven by the MPC8272 or an external SPI device.

The SPI master-in slave-out SPIMISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPIMOSI signal is an output for master devices and an input for slave devices. The dual functionality of these signals allows the SPIs in a multiple-master environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, SPICLK is the clock output signal that shifts received data in from SPIMISO and transmitted data out to SPIMOSI. SPI masters must output a slave select signal to enable SPI slave devices by using a separate general-purpose I/O signal. Assertion of an SPI’s $\overline{\text{SPISEL}}$ while it is master causes an error.
- When the SPI is a slave, SPICLK is the clock input that shifts received data in from SPIMOSI and transmitted data out through SPIMISO. $\overline{\text{SPISEL}}$ is the enable input to the SPI slave. In a multiple-master environment, $\overline{\text{SPISEL}}$ (always an input) is used to detect an error when more than one master is operating.

As described in Chapter 37, “Parallel I/O Ports,” SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$ are multiplexed with port D[16:19] signals, respectively. They are configured as SPI signals through the port D signal assignment register (PDPAR) and the port D data direction register (PDDIR), specifically by setting PDPAR[DD*n*] and PDDIR[DR*n*].

35.3 Configuring the SPI Controller

The SPI can be programmed to work in a single- or multiple-master environment. This section describes SPI master and slave operation in a single-master configuration and then discusses the multi-master environment.

35.3.1 The 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 MPC8272 with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in Figure 35-2. To eliminate the multiple-master error in a single-master environment, the master’s $\overline{\text{SPISEL}}$ input can be forced inactive by selecting port D[19] for general-purpose I/O (PDPAR[DD19] = 0).

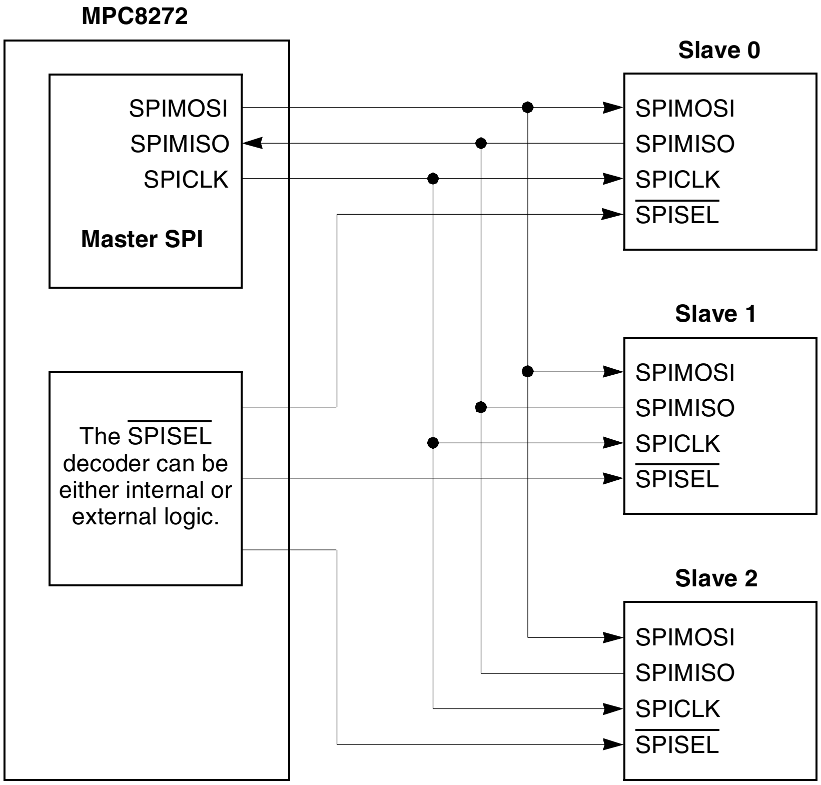


Figure 35-2. Single-Master/Multi-Slave Configuration

To start exchanging data, the core writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] in the SPI command register to start sending data, which starts once the SDMA channel loads the Tx FIFO with data.

The SPI then generates programmable clock pulses on SPICLK for each character and simultaneously shifts Tx data out on SPIMOSI and Rx data in on SPIMISO. Received data is written into a Rx buffer using the next available RxBD. The SPI keeps sending and receiving characters until the whole buffer is sent or an error occurs. The CP follows by clearing TxBD[R] and RxBD[E] and issuing a maskable interrupt to the interrupt controller in the SIU.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically there is no delay on SPIMOSI between buffers. If the current TxBD[L] is set, sending stops after the current buffer is sent. In addition, the RxBD is closed after transmission stops, even if the Rx buffer is not full; therefore, Rx buffers need not be the same length as Tx buffers.

35.3.2 The SPI as a Slave Device

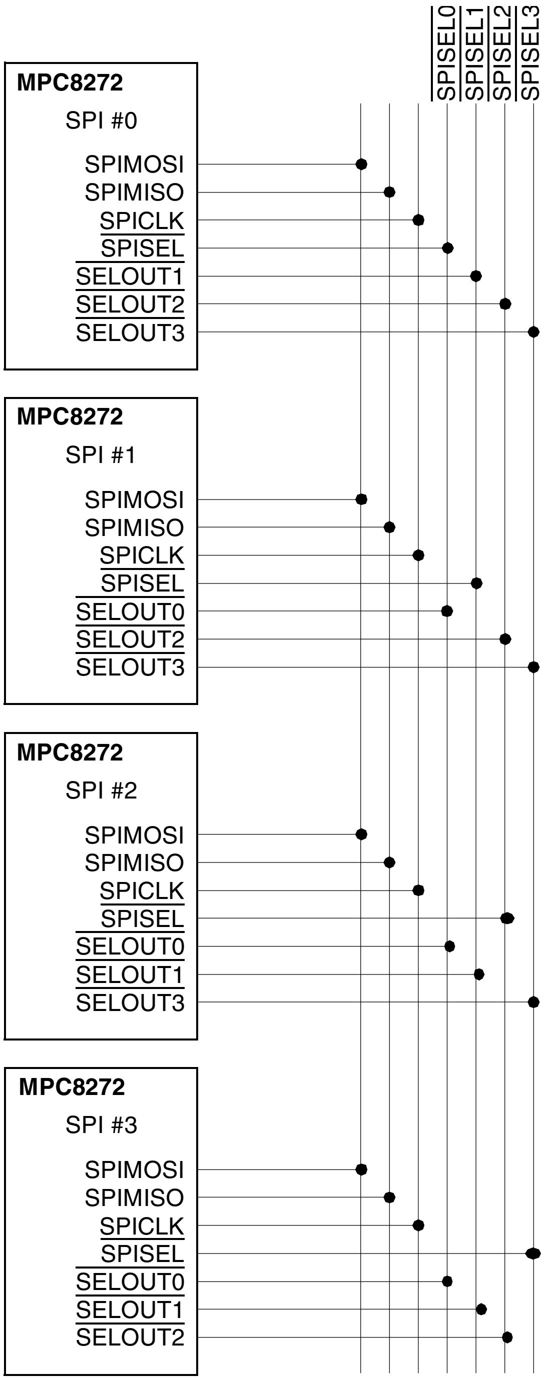
In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's $\overline{\text{SPISEL}}$ must be asserted before Rx clocks are recognized; once $\overline{\text{SPISEL}}$ is asserted, SPICLK becomes an input from the master to the slave. SPICLK can be any frequency from DC to BRGCLK/2 (12.5 MHz for a 25-MHz system).

To prepare for data transfers, the slave's core writes data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] to activate the SPI. Once $\overline{\text{SPISEL}}$ is asserted, the slave shifts data out from SPIMISO and in through SPIMOSI. A maskable interrupt is issued when a full buffer finishes receiving and sending or after an error. The SPI uses successive RxBDs in the table to continue reception until it runs out of Rx buffers or $\overline{\text{SPISEL}}$ is negated.

Transmission continues until no more data is available or $\overline{\text{SPISEL}}$ is negated. If it is negated before all data is sent, it stops but the TxBD stays open. Transmission continues once $\overline{\text{SPISEL}}$ is reasserted and SPICLK begins toggling. After the characters in the buffer are sent, the SPI sends ones as long as $\overline{\text{SPISEL}}$ remains asserted.

35.3.3 The SPI in Multiple-Master Operation

The SPI can operate in a multiple-master environment in which SPI devices are connected to the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK signals of all SPIs are shared; the $\overline{\text{SPISEL}}$ inputs are connected separately, as shown in Figure 35-3. Only one SPI device can act as master at a time—all others must be slaves. When an SPI is configured as a master and its $\overline{\text{SPISEL}}$ input is asserted, a multiple-master error occurs because more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the core. It also disables SPI operation and the output drivers of SPI signals. The core must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and reenable the SPI.



Notes:

- All signals are open-drain.
- For a system with more than two masters, SPISEL and SPIE[MME] do not detect all possible conflicts.
- It is the responsibility of software to arbitrate for the SPI bus (with token passing, for example).
- SELOUT*x* signals are implemented in software with general-purpose I/O signals.

Figure 35-3. Multiple-Master Configuration

The maximum sustained data rate that the SPI supports is SYSTEMCLK/50. However, the SPI can transfer a single character at much higher rates—SYSTEMCLK/4 in master mode and SYSTEMCLK/2 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.

35.4 Programming the SPI Registers

The following sections describe the registers used in configuring and operating the SPI.

35.4.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 35-4, controls both the SPI operation mode and clock source.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN				PM			
Reset	0000_00						—	0_0000_0000								
R/W	R/W															
Addr	0x11AA0															

Figure 35-4. SPMODE—SPI Mode Register

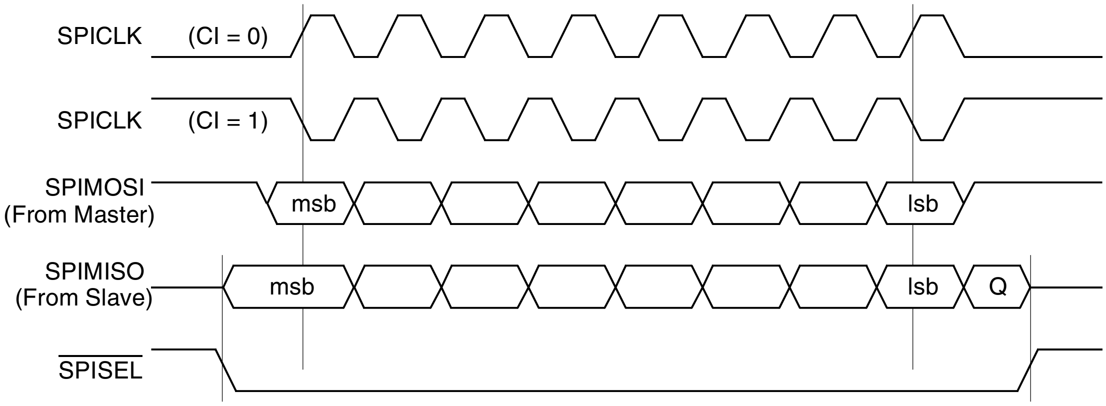
Table 35-1 describes the SPMODE fields.

Table 35-1. 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 35-5 and Figure 35-6. 0 The inactive state of SPICLK is low. 1 The inactive state of SPICLK is high.
3	**CP**	Clock phase. Selects the transfer format. See Figure 35-5 and Figure 35-6. 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 BRGCLK is the input to the SPI BRG. 1 BRGCLK/16 is the input to the SPI BRG.
5	**REV**	Reverse data. Determines the receive and transmit character bit order. 0 Reverse data—lsb of the character sent and received first 1 Normal operation—msb of the character sent and received first
6	**M/S**	Master/slave. Selects master or slave mode. 0 The SPI is a slave. 1 The SPI is a master.

Table 35-1. SPMODE Field Descriptions (continued)

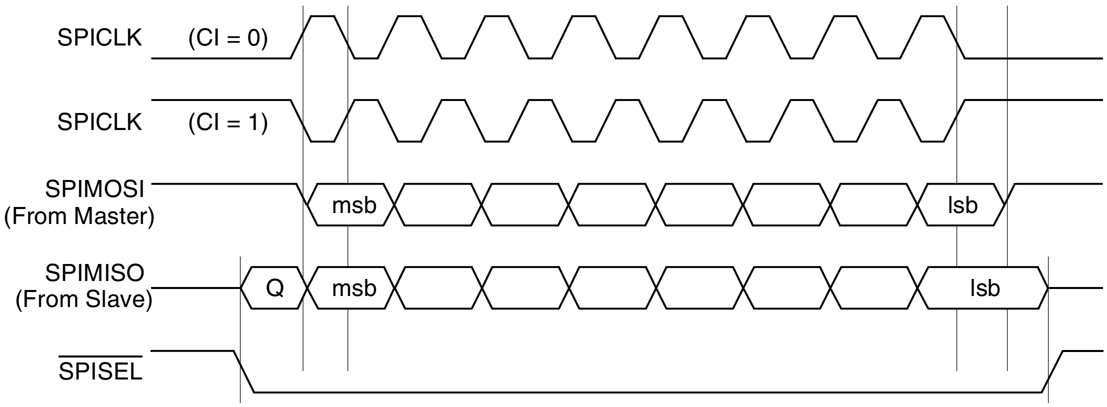
Bits	Name	Description
7	**EN**	Enable SPI. Do not change other SPMODE bits when EN is set. 0 The SPI is disabled. The SPI is in a reset state and consumes minimal power. The SPI BRG is not functioning and the input clock is disabled. 1 The SPI is enabled. Configure SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$ to connect to the SPI as described in Section 37.2, "Port Registers. "
8–11	**LEN**	Character length in bits per character. If the character length is not greater than a byte, every byte in memory holds (LEN+1) valid bits. If the character length is greater than a byte, every half-word holds (LEN+1) valid bits. See Section 35.4.1.1, "SPI Examples with Different SPMODE[LEN] Values." 0000–0010 Reserved, causes erratic behavior. 0011 4-bit characters … 1111 16-bit characters
12–15	**PM**	Prescale modulus select. Specifies the divide ratio of the prescale divider in the SPI clock generator. BRGCLK is divided by 4 * ([PM0–PM3] + 1), a range from 4 to 64. The clock has a 50% duty cycle.



NOTE: Q = Undefined Signal.

Figure 35-5. SPI Transfer Format with SPMODE[CP] = 0

Figure 35-6 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



NOTE: Q = Undefined Signal.

Figure 35-6. SPI Transfer Format with SPMODE[CP] = 1

35.4.1.1 SPI Examples with Different SPMODE[LEN] Values

The examples below show how SPMODE[LEN] is used to determine character length. To help map the process, the conventions shown in Table 35-2 are used in the examples.

Table 35-2. Example Conventions

Convention	Description
g–v	Binary symbols
x	Deleted bit
__[1]	Original byte boundary
_[1]	Original 4-bit boundary.

[1] Both __ and _ are used to aid readability.

Once the data string image is determined, it is always transmitted byte by byte with the lsb of the most-significant byte sent first. For all examples below, assume the memory contains the following binary image:

```
         msb      ghij_klmn__opqr_stuv       lsb
```

Example 1

```
with LEN=4 (data size=5), the following data is selected:
         msb      xxxj_klmn__xxxr_stuv       lsb
with REV=0, the data string image is:
         msb      j_klmn__r_stuv             lsb
the order of the string appearing on the line, a byte at a time is:
         first    nmlk_j__vuts_r             last
with REV=1,the string has each byte reversed, and the data string image is:
         msb      nmlk_j__vuts_r             lsb
the order of the string appearing on the line, one byte at a time is:
         first    j_klmn__r_stuv             last
```

Example 2

```
with LEN=7 (data size=8), the following data is selected:
         msb      ghij_klmn__opqr_stuv       lsb
the data string is selected:
         msb      ghij_klmn__opqr_stuv       lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
         first    nmlk_jihg__vuts_rqpo       last
with REV=1, the string is byte reversed and transmitted, a byte at a time, with lsb
first:
         first    ghij_klmn__opqr_stuv       last
```

Example 3

```
with LEN=0xC (data size=13), the following data is selected:
         msb      ghij_klmn__xxxr_stuv       lsb
the data string selected is:
         msb      r_stuv__ghij_klmn          lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
         first    vuts_r__nmlk_jihg          last
with REV=1, the string is half-word reversed:
         msb      nmlk_jihg__vuts_r          lsb
and transmitted a byte at a time with lsb first:
         first    ghij_klmn__r_stuv          last
```

35.4.2 SPI Event/Mask Registers (SPIE/SPIM)

The SPI event register (SPIE) generates interrupts and reports events recognized by the SPI. When an event is recognized, the SPI sets the corresponding SPIE bit. Clear SPIE bits by writing a 1—writing 0 has no effect. Setting a bit in the SPI mask register (SPIM) enables and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the CP clears internal interrupt requests. Figure 35-7 shows both registers.

	0	1	2	3	4	5	6	7
Field	—		MME	TXE	—	BSY	TXB	RXB
Reset	0000_0000							
R/W	R/W							
Addr	0x11AA6 (SPIE); 0x11AAA (SPIM)							

Figure 35-7. SPIE/SPIM—SPI Event/Mask Registers

Table 35-3 describes the SPIE/SPIM fields.

Table 35-3. SPIE/SPIM Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	MME	Multiple-master error. Set when $\overline{\text{SPISEL}}$ is asserted externally while the SPI is in master mode.
3	TXE	Tx error. Set when an error occurs during transmission. This event is not maskable via the TxBD[I] bit.
4	—	Reserved, should be cleared
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.

Table 35-3. SPIE/SPIM Field Descriptions (continued)

Bits	Name	Description
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Wait two character times to be sure data is completely sent over the transmit signal.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the BD is closed.

35.4.3 SPI Command Register (SPCOM)

The SPI command register (SPCOM), shown in Figure 35-8, is used to start SPI operation.

	0	1	7
Field	STR	—	
Reset	0000_0000		
R/W	Write Only		
Addr	0x11AAD		

Figure 35-8. SPCOM—SPI Command Register

Table 35-4 describes the SPCOM fields.

Table 35-4. SPCOM Field Descriptions

Bits	Name	Description
0	**STR**	Start transmit. For an SPI master, setting STR causes the SPI to start transferring data to and from the Tx/Rx buffers if they are prepared. For a slave, setting STR when the SPI is idle causes it to load the Tx data register from the SPI Tx buffer and start sending with the next SPICLK after $\overline{\text{SPISEL}}$ is asserted. STR is cleared automatically after one system clock cycle.
1–7	—	Reserved and should be cleared

35.5 SPI Parameter RAM

The SPI parameter RAM area is similar to the SCC general-purpose parameter RAM. The CP accesses the SPI parameter table using a user-programmed pointer (SPI_BASE) located in the parameter RAM; see Section 13.5.2, “Parameter RAM.” The SPI parameter table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11 and 12). Some parameter values must be user-initialized before the SPI is enabled; the CP initializes the others. Once initialized, parameter RAM values do not usually need to be accessed. They should be changed only when the SPI is inactive. Table 35-5 shows the memory map of the SPI parameter RAM.

Table 35-5. SPI Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/Tx BD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the SPI. Initialize RBASE/TBASE before enabling the SPI. Furthermore, do not configure BD tables of the SPI to overlap any other active controller’s parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Section 35.5.1, “Receive/Transmit Function Code Registers (RFCR/TFCR).”
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. The SPI has one MRBLR entry to define the maximum number of bytes the MPC8272 writes to a Rx buffer before moving to the next buffer. The MPC8272 can write fewer bytes than MRBLR if an error or end-of-frame occurs, but never exceeds the MRBLR value. User-supplied buffers should be no smaller than MRBLR. Tx buffers are unaffected by MRBLR and can have varying lengths; the number of bytes to be sent is programmed in TxBD[Data Length]. MRBLR is not intended to be changed while the SPI is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the SPI receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state.[2] Reserved for CP use.
0x0C	—	Word	The Rx internal data pointer [2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the current 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 CP initializes RBPTR to the RBASE value. Most applications should not modify RBPTR, but it can be updated when the receiver is disabled or when no Rx buffer is in use.
0x12	—	Hword	The Rx internal byte count [2] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	—	Word	Rx temp.[2] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[2] Reserved for CP use.
0x1C	—	Word	The Tx internal data pointer[2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the current 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 CP initializes TBPTR to the TBASE value. Most applications do not need to modify TBPTR, but it can be updated when the transmitter is disabled or when no Tx buffer is in use.
0x22	—	Hword	The Tx internal byte count[2] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	—	Word	Tx temp.[2] Reserved for CP use.
0x34	—	Word	SDMA temp

[1] From the pointer value programmed in SPI_BASE at IMMR + 0x89FC.

[2] Normally, these parameters need not be accessed. They are listed to help experienced users in debugging.

35.5.1 Receive/Transmit Function Code Registers (RFCR/TFCR)

Figure 35-9 shows the fields in the receive/transmit function code registers (RFCR/TFCR).

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		TC2	DTB	—
Reset	0000_0000							
R/W	R/W							
Addr	SPI Base + 04 (RFCR)/SPI Base + 05 (TFCR)							

Figure 35-9. RFCR/TFCR—Function Code Registers

Table 35-6 describes the RFCR/TFCR fields.

Table 35-6. RFCR/TFCR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	GBL	Global access bit 0 Disable memory snooping 1 Enable memory snooping
3–4	BO	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame or BD. 00 True little-endian. Note this mode can only be used with 32-bit port size memory. 01 Munged little-endian 1x Big-endian
5	TC2	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	DTB	Data bus indicator. 0 Use 60x bus for SDMA operation 1 Reserved
7	—	Reserved, should be cleared

35.6 SPI Commands

Table 35-7 lists transmit/receive commands sent to the CP command register (CPCR).

Table 35-7. SPI Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state and should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

Table 35-7. SPI Commands (continued)

Command	Description
CLOSE RXBD	Forces the SPI controller to close the current RxBD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

35.7 The SPI Buffer Descriptor (BD) Table

As shown in Figure 35-10, BDs are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and SMCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

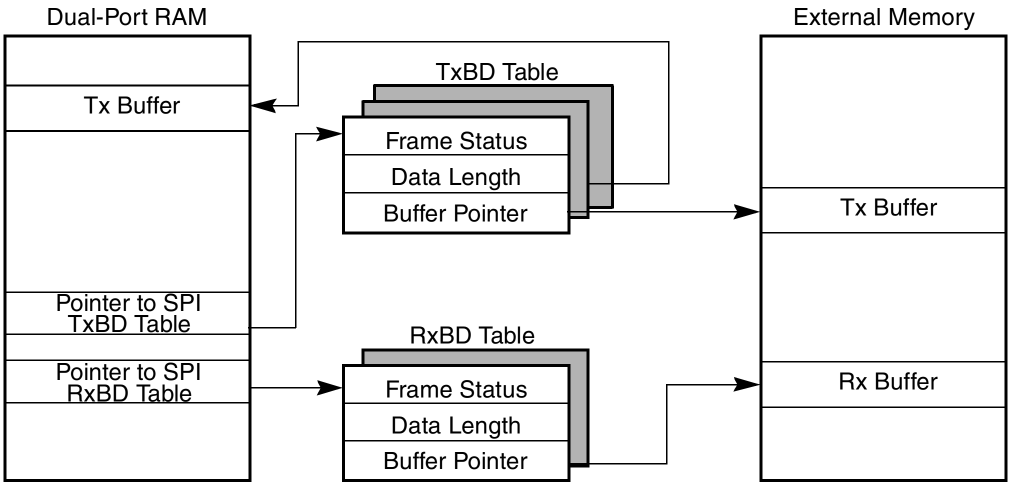


Figure 35-10. SPI Memory Structure

35.7.1 SPI Buffer Descriptors (BDs)

Receive and transmit BDs report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 35-11 and Figure 35-12, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - — For an RxBD, this is the number of octets the CP writes into this RxBD’s buffer once the BD closes. The CP updates this field after the received data is placed into the buffer. Memory allocated for this buffer should be no smaller than MRBLR.

  - — For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. If the character length is more than 8 bits, the data length should be even. For example, to send three characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to send three characters of 9-bit data, the data length field should be initialized to 6 since the three 9-bit data fields occupy three half-words in memory. The CP never modifies this field.
- The word at offset + 4 points to the beginning of the buffer.
  - — For an RxBD, the pointer must be even and can point to internal or external memory.
  - — For a TxBD, the pointer can be even or odd, unless the character exceeds 8 bits, for which it must be even. The buffer can be in internal or external memory.

35.7.1.1 SPI Receive BD (RxBD)

The CP uses RxBDs to report on each received buffer. It closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer once the current buffer is full. The CP also closes the buffer when the SPI is configured as a slave and $\overline{\text{SPISEL}}$ is negated, indicating that reception stopped. The core should write RxBD bits before the SPI is enabled. The format of an RxBD is shown in Figure 35-11.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	—	W	I	L	—	CM	—							OV	ME
Offset + 2	Data Length															
Offset + 4	Rx Buffer Pointer															
Offset + 6																

Figure 35-11. SPI RxBD

Table 35-8 describes the RxBD status and control fields.

Table 35-8. SPI RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared
2	**W**	Wrap (last BD in table). 0 Not the last BD in the RxBD table 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 SPIE[RXB] is set when this buffer is full, indicating the need for the core to process the buffer. SPIE[RXB] causes an interrupt if not masked.

Table 35-8. SPI RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
4	L	Last. Updated by the SPI when the buffer is closed because $\overline{\text{SPISEL}}$ was negated (slave mode only). Otherwise, RxBD[ME] is set. The SPI updates L after received data is placed in the buffer. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.
5	—	Reserved, should be cleared
6	**CM**	Continuous mode. Master mode only; in slave mode, CM should be cleared. 0 Normal operation 1 The CP does not clear RxBD[E] after this BD is closed; the buffer is overwritten when the CP next accesses this BD. This allows continuous reception from an SPI slave into one buffer for autoscanning of a serial A/D peripheral with no core overhead.
7–13	—	Reserved, should be cleared
14	OV	Overrun. Set when a receiver overrun occurs during reception (slave mode only). The SPI updates OV after the received data is placed in the buffer.
15	ME	Multiple-master error. Set when this buffer is closed because $\overline{\text{SPISEL}}$ was asserted when the SPI was in master mode. Indicates a synchronization problem between multiple masters on the SPI bus. The SPI updates ME after the received data is placed in the buffer.

35.7.1.2 SPI Transmit BD (TxBD)

Data to be sent with the SPI is sent to the CP by arranging it in buffers referenced by TxBDs in the TxBD table. TxBD fields should be prepared before data is sent. The format of an TxBD is shown in Figure 35-12.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	—	W	I	L	—	CM	—							UN	ME
Offset + 2	Data Length															
Offset + 4	Tx Buffer Pointer															
Offset + 6																

Figure 35-12. SPI TxBD

Table 35-9 describes the TxBD status and control fields.

Table 35-9. 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 CP clears R (unless RxBD[CM] is set) after the buffer is sent (unless RxBD[CM] is set) or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved, should be cleared

Table 35-9. SPI TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is processed; SPIE[TXE] is unaffected. 1 SPIE[TXB] or SPIE[TXE] are set when this buffer is processed and causes interrupts if not masked.
4	**L**	Last. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.
5	—	Reserved, should be cleared
6	**CM**	Continuous mode. Valid only when the SPI is in master mode. In slave mode, it should be cleared. 0 Normal operation 1 The CP does not clear TxBD[R] after this BD is closed, allowing the buffer to be resent automatically when the CP next accesses this BD.
7–13	—	Reserved, should be cleared
14	UN	Underrun. Indicates that the SPI encountered a transmitter underrun condition while sending the buffer. This error occurs only when the SPI is in slave mode. The SPI updates UN after it sends the buffer.
15	ME	Multiple-master error. Indicates that this buffer is closed because 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.

35.8 SPI Master Programming Example

The following sequence initializes the SPI to run at a high speed in master mode:

1. Configure port D to enable SPIMISO, SPIMOSI, SPICLK and $\overline{\text{SPISEL}}$.
2. Configure a parallel I/O signal to operate as the SPI select output signal if needed.
3. In address 0x89FC, assign a pointer to the SPI parameter RAM.
4. Write RBASE and TBASE in the SPI parameter RAM to point to the RxBD and TxBD tables in the dual-port RAM. Assuming one RxBD followed by one TxBD at the beginning of the dual-port RAM, write RBASE with 0x0000 and TBASE with 0x0008.
5. Write RFCR and TFCR with 0x10 for normal operation.
6. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
7. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].

8. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
9. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
10. Write 0xFF to SPIE to clear any previous events.
11. Write 0x37 to SPIM to enable all possible SPI interrupts.
12. Write 0x0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
13. Set SPCOM[STR] to start the transfer.

After 5 bytes are sent, the TxBD is closed. Additionally, the Rx buffer is closed after 5 bytes are received because TxBD[L] is set.

35.9 SPI Slave Programming Example

The following is an example initialization sequence to follow when the SPI is in slave mode. It is very similar to the SPI master example, except that $\overline{\text{SPISEL}}$ is used instead of a general-purpose I/O signal (as shown in Figure 35-2).

1. Enable SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$.
2. In address 0x89FC, assign a pointer to the SPI parameter RAM.
3. Assuming one RxBD at the beginning of the dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008 in the SPI parameter RAM.
4. Write RFCR and TFCR with 0x08 for normal operation.
5. Program MRBLR = 0x0010 for 16 bytes, the maximum number of bytes per buffer.
6. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
7. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
8. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
9. Write 0xFF to SPIE to clear any previous events.
10. Write 0x37 to SPIM to enable all SPI interrupts.
11. Set SPMODE to 0x0170 to enable normal operation (not loopback), slave mode, SPI enabled, and 8-bit characters. BRG speed is ignored in slave mode.
12. Set SPCOM[STR] to enable the SPI to be ready once the master begins the transfer.

NOTE

If the master sends 3 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed but the TxBD remains open. If the master sends 5 or more bytes, the TxBD is closed after the fifth byte. If the master sends 16 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed without triggering an out-of-buffers error. If the master sends more than 16 bytes, the RxBD is closed (full) and an out-of-buffers error occurs after the 17th byte is received.

35.10 Handling Interrupts in the SPI

The following sequence should be followed to handle interrupts in the SPI:

1. Once an interrupt occurs, read SPIE to determine the interrupt source. Normally, SPIE bits should be cleared at this time.
2. Process the TxBD to reuse it and the RxBD to extract the data from it. To transmit another buffer, simply set TxBD[R], RxBD[E], and SPCOM[STR].
3. Execute an **rfi** instruction.

Chapter 36
I²C Controller

The inter-integrated circuit (I^{2}C®) controller lets the MPC8272 exchange data with other I^{2}C devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCD displays. The I^{2}C controller uses a synchronous, multiple-master bus that can connect several integrated circuits on a board. It uses two signals—serial data (SDA) and serial clock (SCL)—to carry information between the integrated circuits connected to it.

As shown in Figure 36-1, the I^{2}C controller consists of transmit and receive sections, an independent baud-rate generator (BRG), and a control unit. The transmit and receive sections use the same clock, which is derived from the I^{2}C BRG when in master mode and generated externally when in slave mode. Wait states are inserted during a data transfer if SCL is held low by a slave device. In the middle of a data transfer, the master I^{2}C controller recognizes the need for wait states by monitoring SCL. However, the I^{2}C controller has no automatic time-out mechanism if the slave device does not release SCL; therefore, software should monitor how long SCL stays low to generate bus timeouts.

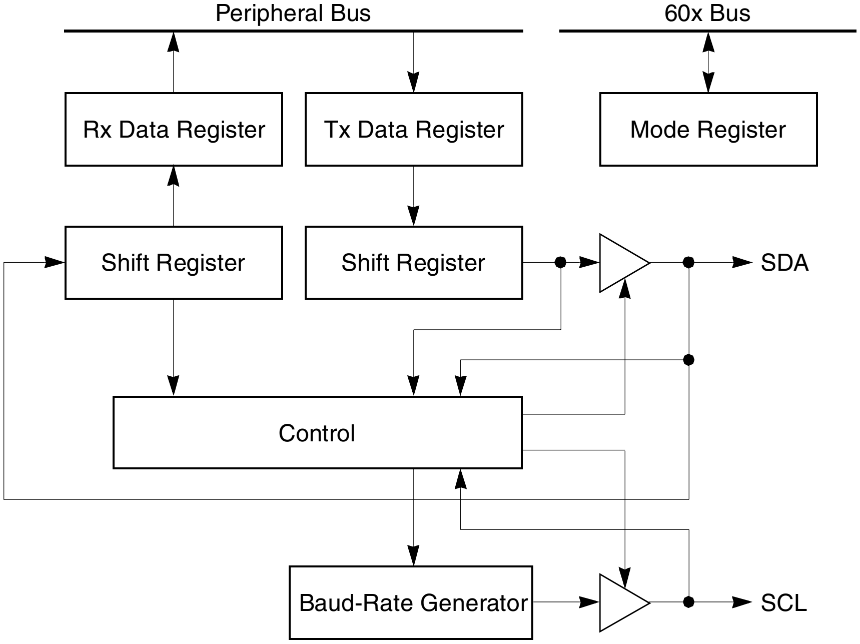


Figure 36-1. I^{2}C Controller Block Diagram

The I^{2}C receiver and transmitter are double-buffered, which corresponds to an effective two-character FIFO latency. In normal operation, the transmitter shifts the msb (bit 0) out first. When the I^{2}C is not enabled in the I^{2}C mode register (I2MOD[EN] = 0), it consumes little power.

36.1 Features

The following is a list of the I^{2}C controller’s main features:

- Two-signal interface (SDA and SCL)
- Supports master and slave I^{2}C operation
- Supports multiple-master environment
- Continuous transfer mode for automatic scanning of a peripheral
- Supports a maximum clock rate of 2,080 KHz (with a CPM utilization of 25%), assuming a 100-MHz system clock
- Independent, programmable baud-rate generator
- Supports 7-bit I^{2}C addressing
- Open-drain output signals allow multiple master configuration
- Local loopback capability for testing

36.2 I^{2}C Controller Clocking and Signal Functions

The I^{2}C controller can be configured as a master or slave for the serial channel. As a master, the controller’s BRG provides the transfer clock. The I^{2}C BRG takes its input from the BRG clock (BRGCLK), which is generated from the CPM clock; see Section 10.4, “System Clock Control Register (SCCR).”

SDA and SCL are bidirectional signals connected to a positive supply voltage through an external pull-up resistor. When the bus is free, both signals are pulled high. The general I^{2}C master/slave configuration is shown in Figure 36-2.

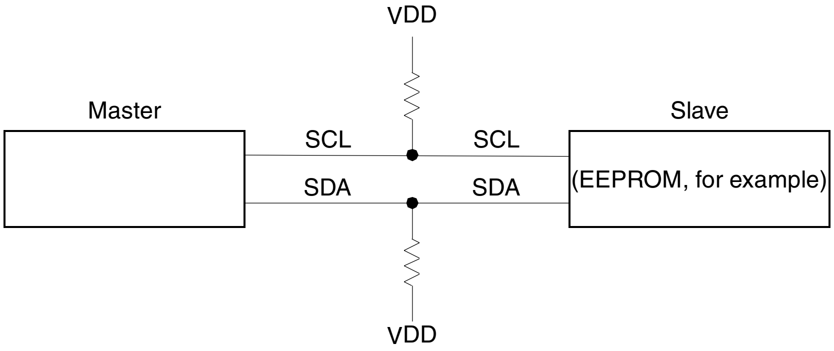


Figure 36-2. I^{2}C Master/Slave General Configuration

When the I^{2}C controller is master, the SCL clock output, taken directly from the I^{2}C BRG, shifts receive data in and transmit data out through SDA. The transmitter arbitrates for the bus during transmission and aborts if it loses arbitration. When the I^{2}C controller is a slave, the SCL clock input shifts data in and out through SDA. The SCL frequency can range from DC to BRGCLK/48.

36.3 I^{2}C Controller Transfers

To initiate a transfer, the master I^{2}C controller sends a message specifying a read or write request to an I^{2}C slave. The first byte of the message consists of a 7-bit slave port address and a R/W request bit. Note that because the R/W request follows the slave port address in the I^{2}C bus specification, the R/W request bit must be placed in the lsb (bit 7) unless operating in reverse data mode; see Section 36.4.1, "I^{2}C Mode Register (I2MOD)."

To write to a slave, the master sends a write request (R/W = 0) along with either the target slave's address or a general call (broadcast) address of all zeros, followed by the data to be written. To read from a slave, the master sends a read request (R/W = 1) and the target slave's address. When the target slave acknowledges the read request, the transfer direction is reversed, and the master receives the slave's transmit buffer(s). If the receiver (master or slave) does not acknowledge each byte transfer in the ninth bit frame, the transmitter signals a transmission error event (I2ER[TXE]). An I^{2}C transfer timing diagram is shown in Figure 36-3.

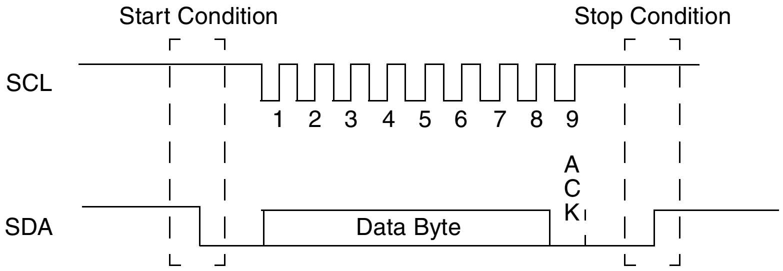


Figure 36-3. I^{2}C Transfer Timing

Select master or slave mode for the controller using the I^{2}C command register (I2COM[M/S]). Set the master's start bit, I2COM[STR], to begin a transfer; setting a slave's I2COM[STR] activates the slave to wait for a transfer request from a master.

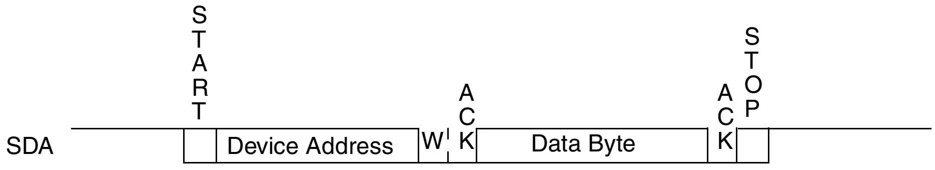
If a master or slave transmitter's current TxBD[L] is set, transmission stops once the buffer is sent; that is, I2COM[STR] must be set again to reactivate transfers. If TxBD[L] is zero, once the current buffer is sent, the controller begins processing the next TxBD without waiting for I2COM[STR] to be set again.

The following sections further detail the transfer process.

36.3.1 I^{2}C Master Write (Slave Read)

If the MPC8272 is the master, prepare the transmit buffers and BDs before initiating a write. Initialize the first transmit data byte with the slave address and write request (R/W = 0).

If the MPC8272 is the slave target of the write, prepare receive buffers and BDs to await the master's request. Figure 36-4 shows the timing for a master write.



Note: Data and ACK are repeated n times.

Figure 36-4. I^{2}C Master Write Timing

A master write occurs as follows:

1. The master core sets I2COM[STR]. The transfer starts when the SDMA channel loads the Tx FIFO with data and the I^{2}C bus is not busy.
2. The I^{2}C master generates a start condition—a high-to-low transition on SDA while SCL is high—and the transfer clock SCL pulses for each bit shifted out on SDA. If the master transmitter detects a multiple-master collision (by sensing a ‘0’ on SDA while sending a ‘1’), transmission stops and the channel reverts to slave mode. A maskable interrupt is sent to the master’s core so software can try to retransmit later.
3. The slave acknowledges each byte and writes to its current receive buffer until a new start or stop condition is detected.
4. After sending each byte, the master monitors the acknowledge indication. If the slave receiver fails to acknowledge a byte, transmission stops and the master generates a stop condition—a low-to-high transition on SDA while SCL is high.

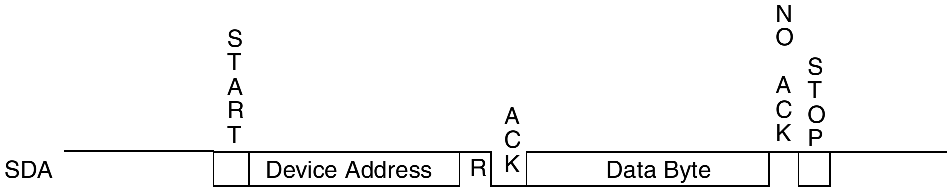
36.3.2 I^{2}C Loopback Testing

When in master mode, an I^{2}C controller supports loopback operation for master write requests. The master I^{2}C controller simply issues a write request directed to its own address (programmed in I2ADD). The master’s receiver monitors the transmission and reads the transmitted data into its receive buffer. Loopback operation requires no special register programming.

36.3.3 I^{2}C Master Read (Slave Write)

Before initiating a master read with the MPC8272, prepare a transmit buffer of size *n* + 1 bytes, where *n* is the number of bytes to be read from the slave. The first transmit byte should be initialized to the slave address with R/W = 1. The next *n* transmit bytes are used strictly for timing and can be left uninitialized. Configure suitable receive buffers and BDs to receive the slave’s transmission.

If the MPC8272 is the slave target of the read, prepare the I^{2}C transmit buffers and BDs and activate it by setting I2COM[STR]. Figure 36-5 shows the timing for a master read.



Note: After the nth data byte, the master does not acknowledge the slave.

Figure 36-5. I^{2}C Master Read Timing

A master read occurs as follows:

1. Set the master’s I2COM[STR] to initiate the read. The transfer starts when the SDMA channel loads the transmit FIFO with data and the I^{2}C bus is not busy.
2. The slave detects a start condition on SDA and SCL.

3. After the first byte is shifted in, the slave compares the received data to its slave address. If the slave is an MPC8272, the address is programmed in its I^{2}C address register (I2ADD).
   - If a match is found, the slave acknowledges the received byte and begins transmitting on the clock pulse immediately following the acknowledge.
   - If a match is found but the slave is not ready, the read request is not acknowledged and the transaction is aborted. If the slave is an MPC8272, a maskable transmission error interrupt is triggered to allow software to prepare data for transmission on the next try.
   - If a mismatch occurs, the slave ignores the message and searches for a new start condition.
4. The master acknowledges each byte sent as long as an overrun does not occur. If the master receiver fails to acknowledge a byte, the slave aborts transmission. For a slave MPC8272, the abort generates a maskable interrupt. A maskable interrupt is also issued after a complete buffer is sent or after an error. If an underrun occurs, the MPC8272 slave sends ones until a stop condition is detected.

36.3.4 I^{2}C Multiple-Master Considerations

The I^{2}C controller supports a multiple-master configuration, in which the I^{2}C controller must alternate between master and slave modes. The I^{2}C controller supports this by implementing I^{2}C master arbitration in hardware. However, due to the nature of the I^{2}C bus and the implementation of the I^{2}C controller, certain software considerations must be made.

A MPC8272 I^{2}C controller attempting a master read request could simultaneously be targeted for an external master write (slave read). Both operations trigger the controller’s I2CER[RXB] event, but only one operation wins the bus arbitration. To determine which operation caused the interrupt, software must verify that its transmit operation actually completed before assuming that the received data is the result of its read operation.

Problems could also arise if the MPC8272's I^{2}C controller master sets up a transmit buffer and BD for a write request, but then is the target of a read request from another master. Without software precautions, the I^{2}C controller responds to the other master with the transmit buffer originally intended for its own write request. To avoid this situation, a higher-level handshake protocol must be used. For example, a master, before reading a slave, writes the slave with a description of the requested data (which register should be read, for example). This operation is typical with many I^{2}C devices.

In addition, it is not recommended to enable the PowerQUICC II’s I^{2}C controller while another I^{2}C master is executing transactions on the bus. The PowerQUICC II’s I^{2}C controller should wait for the bus to become idle.

The PowerQUICC II’s I^{2}C controller assumes that other I^{2}C devices on the bus closely conform to the I^{2}C specification. Unexpected behavior can occur if the PowerQUICC II I^{2}C controller is connected with devices which operate outside the specification. For example, a slave device which acknowledges a master write with two SCL pulses instead of one (total of 10 SCL pulses), can cause wrong behavior of the PowerQUICC I^{2}C controller on its next transaction.

36.4 I^{2}C Registers

The following sections describe the I^{2}C registers.

36.4.1 I^{2}C Mode Register (I2MOD)

The I^{2}C mode register, shown in Figure 36-6, controls the I^{2}C modes and clock source.

	0	1	2	3	4	5	6	7
Field	—		REVD	GCD	FLT	PDIV		EN
Reset	0000_0000							
R/W	R/W							
Addr	0x11860							

Figure 36-6. I^{2}C Mode Register (I2MOD)

Table 36-1 describes I2MOD bit functions.

Table 36-1. I2MOD Field Descriptions

Bits	Name	Description
0–1	—	Reserved and should be cleared
2	REVD	Reverse data. Determines the Rx and Tx character bit order. 0 Normal operation. The msb (bit 0) of a character is transferred first. 1 Reverse data. the lsb (bit 7) of a character is transferred first. **Note:** Clearing REVD is strongly recommended to ensure consistent bit ordering across devices.
3	GCD	General call disable. Determines whether the receiver acknowledges a general call address. 0 General call address is enabled 1 General call address is disabled
4	FLT	Clock filter. Determines if the I^{2}C input clock SCL is filtered to prevent spikes in a noisy environment. 0 SCL is not filtered. 1 SCL is filtered by a digital filter.
5–6	PDIV	Predivider. Selects the clock division factor before it is input into the I^{2}C BRG. The clock source for the I^{2}C BRG is the BRGCLK generated from the CPM clock; see Section 10.4, “System Clock Control Register (SCCR).” 00 BRGCLK/32 01 BRGCLK/16 10 BRGCLK/8 11 BRGCLK/4 To both save power and reduce noise susceptibility, select the PDIV with the largest division factor (slowest clock) that still meets performance requirements.
7	EN	Enable I^{2}C operation. 0 I^{2}C is disabled. The I^{2}C is in a reset state and consumes minimal power. 1 I^{2}C is enabled. Do not change other I2MOD bits when EN is set.

36.4.2 I^{2}C Address Register (I2ADD)

The I^{2}C address register, shown in Figure 36-7, holds the address for this I^{2}C port.

	0 … 6	7
Field	SAD	—
Reset	Undefined	
R/W	R/W	
Addr	0x11864	

Figure 36-7. I^{2}C Address Register (I2ADD)

Table 36-2 describes I2ADD fields.

Table 36-2. I2ADD Field Descriptions

Bits	Name	Description
0–6	SAD	Slave address 0–6. Holds the slave address for the I^{2}C port.
7	—	Reserved and should be cleared.

36.4.3 I^{2}C Baud Rate Generator Register (I2BRG)

The I^{2}C baud rate generator register, shown in Figure 36-8, sets the divide ratio of the I^{2}C BRG.

	0 … 7
Field	DIV
Reset	1111_1111
R/W	R/W
Addr	0x11868

Figure 36-8. I^{2}C Baud Rate Generator Register (I2BRG)

Table 36-3 describes I2BRG fields.

Table 36-3. I2BRG Field Descriptions

Bits	Name	Description
0–7	DIV	Division ratio 0–7. Specifies the divide ratio of the BRG divider in the I^{2}C clock generator. The output of the prescaler is divided by 2 × ([DIV0–DIV7] + 3 + (2 × I2MOD[FLT])) and the clock has a 50% duty cycle. DIV must be programmed to a minimum value of 3 if the digital filter is disabled (I2MOD[FLT] = 0) and 6 if it is enabled (I2MOD[FLT] = 1).

36.4.4 I^{2}C Event/Mask Registers (I2CER/I2CMR)

The I^{2}C event register (I2CER) is used to generate interrupts and report events. When an event is recognized, the I^{2}C controller sets the corresponding I2CER bit. I2CER bits are cleared by writing ones; writing zeros has no effect. Setting a bit in the I^{2}C mask register (I2CMR) enables and clearing a bit masks the corresponding interrupt. Unmasked I2CER bits must be cleared before the CP clears internal interrupt requests. Figure 36-9 shows both registers.

	0		2	3	4	5	6	7
Field	—			TXE	—	BSY	TXB	RXB
Reset	0000_0000							
R/W	R/W							
Addr	0x11870(I2CER)/0x11874 I2CMR)							

Figure 36-9. I^{2}C Event/Mask Registers (I2CER/I2CMR)

Table 36-4 describes the I2CER/I2CMR fields.

Table 36-4. I2CER/I2CMR Field Descriptions

Bits	Name	Description
0–2	—	Reserved and should be cleared
3	TXE	Tx error. Set when an error occurs during transmission. This event is not maskable via the TxBD[I] bit.
4	—	Reserved and should be cleared
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Two character times must elapse to guarantee that all data has been sent.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the RxBD is closed.

36.4.5 I^{2}C Command Register (I2COM)

The I^{2}C command register, shown in Figure 36-10, is used to start I^{2}C transfers and to select master or slave mode.

	0	1	2	3	4	5	6	7
Field	STR	—						M/S
Reset	0000_0000							
R/W	R/W							
Addr	0x1186C							

Figure 36-10. I^{2}C Command Register (I2COM)

Table 36-5 describes I2COM fields.

Table 36-5. I2COM Field Descriptions

Bits	Name	Description
0	STR	Start transmit. In master mode, setting STR causes the I^{2}C controller to start sending data from the I^{2}C Tx buffers if they are ready. In slave mode, setting STR when the I^{2}C controller is idle causes it to load the Tx data register from the I^{2}C Tx buffer and start sending when it receives an address byte that matches the slave address with R/W = 1. STR is always read as a 0.

Table 36-5. I2COM Field Descriptions (continued)

Bits	Name	Description
1–6	—	Reserved and should be cleared
7	M/S	Master/slave. Configures the I^{2}C controller to operate as a master or a slave. 0 I^{2}C is a slave. 1 I^{2}C is a master.

36.5 I^{2}C Parameter RAM

The I^{2}C controller parameter table is used for the general I^{2}C parameters and is similar to the SCC general-purpose parameter RAM. The CP accesses the I^{2}C parameter table using a user-programmed pointer (I2C_BASE) located in the parameter RAM; see Section 13.5.2, “Parameter RAM.” The I^{2}C parameter table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11 and 12). The user must initialize certain parameter RAM values before the I^{2}C is enabled; the CP initializes the other values. Software usually does not access parameter RAM entries once they are initialized; they should be changed only when the I^{2}C is inactive.

Table 36-6 shows the I^{2}C parameter memory map.

Table 36-6. I^{2}C Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the I^{2}C. Initialize RBASE/TBASE before enabling the I^{2}C. Furthermore, do not configure BD tables of the I^{2}C to overlap any other active controller’s parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Figure 36-11 and Table 36-7.
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8272 writes to a Rx buffer before moving to the next buffer. The MPC8272 writes fewer bytes to the buffer than the MRBLR value if an error or end-of-frame occurs. Buffers should not be smaller than MRBLR. Tx buffers are unaffected by MRBLR and can vary in length; the number of bytes to be sent is specified in TxBD[Data Length]. MRBLR is not intended to be changed while the I^{2}C is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the I^{2}C receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state.[2] Reserved for CP use.
0x0C	RPTR	Word	Rx internal data pointer[2] is updated by the SDMA channels to show the next address in the buffer to be accessed.

Table 36-6. I^{2}C Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x10	RBPTR	Hword	RxBD pointer. Points to the next descriptor the receiver transfers data to when it is in an idle state or to the current descriptor during frame processing for each I^{2}C channel. After a reset or when the end of the descriptor table is reached, the CP initializes RBPTR to the value in RBASE. Most applications should not write RBPTR, but it can be modified when the receiver is disabled or when no receive buffer is used.
0x12	RCOUNT	Hword	Rx internal byte count [2] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	RTEMP	Word	Rx temp.[2] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[2] Reserved for CP use.
0x1C	TPTR	Word	Tx internal data pointer [2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the next descriptor that the transmitter transfers data from when it is in an idle state or to the current descriptor during frame transmission. After a reset or when the end of the descriptor table is reached, the CP initializes TBPTR to the value in TBASE.Most applications should not write TBPTR, but it can be modified when the transmitter is disabled or when no transmit buffer is used.
0x22	TCOUNT	Hword	Tx internal byte count [2] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	TTEMP	Word	Tx temp.[2] Reserved for CP use.
0x34	SDMATMP	Word	SDMA temp.[2] Reserved for CP use.

[1] From the pointer value programmed in I2C_BASE at IMMR + 0x8AFC.

[2] Normally, these parameters need not be accessed.

Figure 36-11 shows the RFCR/TFCR bit fields.

	0	1	2	3	4	5	6	7
Field			GBL	**BO**		**TC2**	**DTB**	**—**
Reset	0000_0000							
R/W	R/W							
Addr	I2C_BASE + 04 (RFCR)/I2C_BASE + 05 (TFCR)							

Figure 36-11. I^{2}C Function Code Registers (RFCR/TFCR)

Table 36-7 describes the RFCR/TFCR bit fields.

Table 36-7. RFCR/TFCR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared
2	**GBL**	Global access bit 0 Disable memory snooping 0 Enable memory snooping

Table 36-7. RFCR/TFCR Field Descriptions (continued)

Bits	Name	Description
3–4	**BO**	Byte ordering. Selects the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame or BD. 00 True little-endian. Note this mode can only be used with 32-bit port size memory. 01 Munged little-endian 1x Big-endian
5	**TC2**	Transfer code 2. Contains the transfer code value of TC[2], used during this SDMA channel memory access. TC[0–1] is driven with a 0b11 to identify this SDMA channel access as a DMA-type access.
6	**DTB**	Data bus indicator. 0 Use 60x bus for SDMA operation 1 Reserved
7	—	Reserved, should be cleared

36.6 I^{2}C Commands

The I^{2}C transmit and receive commands, shown in Table 36-8, are issued to the CP command register (CPCR).

Table 36-8. I^{2}C Transmit/Receive Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state. Should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.
CLOSE RXBD	Forces the I^{2}C controller to close the current Rx BD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

36.7 The I^{2}C Buffer Descriptor (BD) Table

As shown in Figure 36-12, buffer descriptors (BDs) are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and SMCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

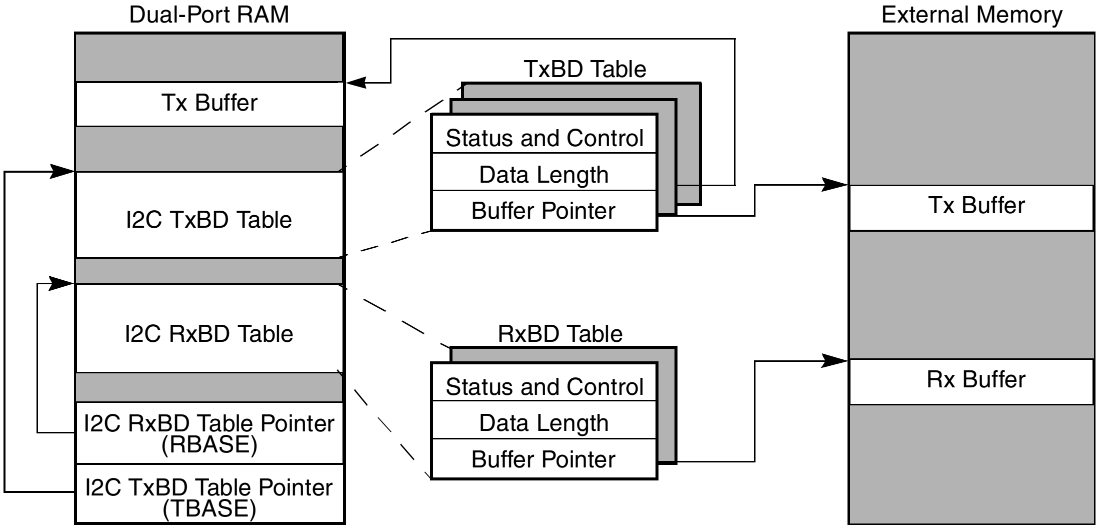


Figure 36-12. I^{2}C Memory Structure

36.7.1 I^{2}C Buffer Descriptors (BDs)

Receive and transmit buffer descriptors report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 36-13 and Figure 36-14, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - For an RxBD, this is the number of octets the CP writes into this RxBD's buffer once the descriptor closes. The CP updates this field after the received data is placed into the associated buffer. Memory allocated for this buffer should be no smaller than MRBLR.
  - For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. The CP never modifies this field.
- The word at offset + 4 points to the beginning of the buffer.
  - For an RxBD, the pointer must be even and can point to internal or external memory.
  - For a TxBD, the pointer can be even or odd. The buffer can reside in internal or external memory.

36.7.1.1 I^{2}C Receive Buffer Descriptor (RxBD)

Using RxBDs, the CP reports on each buffer received, closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer when the current one is full. It closes the buffer when a stop or start condition is found on the I^{2}C bus or when an overrun error occurs. The core should write RxBD bits before the I^{2}C controller is enabled.

	0	1	2	3	4	5	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	—		OV	—
Offset + 2	Data Length								
Offset + 4	**RX Buffer Pointer**								
Offset + 6									

Figure 36-13. I²C RxBD

Table 36-9 describes I²C RxBD status and control bits.

Table 36-9. I²C RxBD Status and Control Bits

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved and should be cleared
2	**W**	Wrap (last BD in table). 0 Not the last BD in the RxBD table 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is full. 1 The I2CER[RXB] is set when the CP fills this buffer, indicating that the core needs to process the buffer. The RXB bit can cause an interrupt if it is enabled.
4	L	Last. The I²C controller sets L. 0 This buffer does not contain the last character of the message. 1 This buffer holds the last character of the message. The I²C controller sets L after all received data is placed into the associated buffer, or because of a stop or start condition or an overrun.
5–13	—	Reserved and should be cleared
14	OV	Overrun. Set when a receiver overrun occurs during reception. The I²C controller updates this bit after the received data is placed into the associated buffer.
15	—	Reserved and should be cleared

36.7.1.2 I²C Transmit Buffer Descriptor (TxBD)

Transmit data is arranged in buffers referenced by TxBDs in the TxBD table. The first word of the TxBD, shown in Figure 36-14, contains status and control bits.

	0	1	2	3	4	5	6	12	13	14	15
Offset + 0	**R**	—	**W**	**I**	**L**	**S**	—		NAK	UN	CL
Offset + 2	**Data Length**										
Offset + 4	**Tx Buffer Pointer**										
Offset + 6											

Figure 36-14. I^{2}C TxBD

Table 36-10 describes I^{2}C TxBD status and control bits.

Table 36-10. I^{2}C TxBD Status and Control Bits

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The CP clears R after the buffer is sent or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved and should be cleared
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table 1 Last BD in the table. After this buffer is used, the CP transmits data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is serviced; I2CER[TXE] is unaffected. 1 I2CER[TXB] or I2CER[TXE] is set when the buffer is serviced. If enabled, an interrupt occurs.
4	**L**	Last. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message. The I^{2}C controller generates a stop condition after sending this buffer.
5	**S**	Generate start condition. Provides ability to send back-to-back frames with one I2COM[STR] trigger. 0 Do not send a start condition before the first byte of the buffer. 1 Send a start condition before the first byte of the buffer. (Used to separate frames.) Note: If this BD is the first one in the frame when I2COM[STR] is triggered, a start condition is sent regardless of the value of TxBD[S].
6–12	—	Reserved and should be cleared
13	NAK	No acknowledge. Indicates that the transmission was aborted because the last byte sent was not acknowledged. The I^{2}C controller updates NAK after the buffer is sent.
14	UN	Underrun. Indicates that the I^{2}C controller encountered a transmitter underrun condition while sending the associated buffer. The I^{2}C controller updates UN after the buffer is sent.
15	CL	Collision. Indicates that transmission terminated because the transmitter was lost while arbitrating for the bus. The I^{2}C controller updates CL after the buffer is sent.

Chapter 37
Parallel I/O Ports

The CPM supports four general-purpose I/O ports—ports A, B, C, and D. Each pin in the I/O ports can be configured as a general-purpose I/O signal or as a dedicated peripheral interface signal. Port C is unique in that 16 of its pins (PC[0:1,4:15,23,29]) can generate interrupts to the interrupt controller.

Each pin can be configured as an input or output and has a latch for data output, read or written at any time. Each pin can be configured as general-purpose I/O or as a dedicated peripheral pin. Some of the pins can be configured as open-drain (in a wired-OR configuration on the board). These pins drives a zero voltage but three-states when driving a high voltage.

Note that port pins do not have internal pull-up resistors. Due to the CPM's significant flexibility, many dedicated peripheral functions are multiplexed onto the ports. The functions are grouped to maximize the pins' usefulness in the greatest number of applications. Note that to obtain a full understanding of the pin assignment capability described in this chapter, a user must understand the CPM peripherals.

37.1 Features

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

- Port A is 24 bits
- Port B is 14 bits
- Port C is 28 bits
- Port D is 16 bits
- All ports are bidirectional.
- All ports have alternate on-chip peripheral functions (however, on the MCP8248 PA[22–23] can be used only as general purpose).
- All ports are three-stated at system reset.
- All pin values can be read while the pin is connected to an on-chip peripheral
- Open-drain capability on some pins
- Port C offers 16 interrupt input pins (PC[0:1,4:15,23,29]).

37.2 Port Registers

Each port has four memory-mapped, read/write, 32-bit control registers.

37.2.1 Port Open-Drain Registers (PODR*x*)

The port open-drain registers (PODR*x*), shown in Figure 37-1, Figure 37-2, Figure 37-3 and Figure 37-4 indicate a normal or wired-OR configuration of the port pins.

37.2.1.1 Port A Open-Drain Register (PODRA)

	0	7	8	9	10	11	12	13	14	15
Field	—		OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	0x10D0C									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	OD30	OD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D0E															

Figure 37-1. Port A Open-Drain Registers (PODRA)

Table 37-1 describes PODRA fields.

Table 37-1. PODRA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared
8–31	OD*x*	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.

37.2.1.2 Port B Open-Drain Register (PODRB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10D2C	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	OD30	OD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D2E															

Figure 37-2. Port B Open-Drain Registers (PODRB)

Table 37-2 describes PODRB fields.

Table 37-2. PODRB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18–31	OD*x*	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.

37.2.1.3 Port C Open-Drain Register (PODRC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OD0	OD1	—		OD4	OD5	OD6	OD7	OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D4C															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D4E															

Figure 37-3. Port C Open-Drain Registers (PODRC)

Table 37-3 describes PODRC fields.

Table 37-3. PODRC Field Descriptions

Bits	Name	Description
0–1	OD*x*	Open-drain configuration. Determines whether the corresponding pin is actively driven as an output or is an open-drain driver. 0 The I/O pin is actively driven as an output. 1 The I/O pin is an open-drain driver. As an output, the pin is driven active-low, otherwise it is three-stated.
2–3	—	Reserved, should be cleared
4–29	OD*x*	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.
30–31	—	Reserved, should be cleared

37.2.1.4 Port D Open-Drain Register (PODRD)

	0	6	7	8	13	14	15
Field	—		OD7	—		OD14	OD15
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10D0C (PODRA), 0x10D2C (PODRB), 0x10D4C (PODRC), 0x10D6C (PODRD)						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	—		OD29	OD30	OD31
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x10D0E (PODRA), 0x10D2E (PODRB), 0x10D4E (PODRC), 0x10D6E (PODRD)														

Figure 37-4. Port D Open-Drain Registers (PODRD)

Table 37-4 describes PODRD fields.

Table 37-4. PODRD Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared
7	OD*7*	Open-drain configuration. Determines whether the PD[7] is actively driven as an output or is an open-drain driver. 0 PD[7] is actively driven as an output. 1 PD[7] is an open-drain driver. As an output, PD[7] is driven active-low; otherwise it is three-stated.
8–13	—	Reserved, should be cleared
14–25	OD*x*	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.
26–28	—	Reserved, should be cleared
29–31	OD*x*	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.

37.2.2 Port Data Registers (PDAT*x*)

A read of port data registers (PDAT*x*), shown in Figure 37-5, Table 37-6, Table 37-7 and Table 37-8, returns the data at the pin, independent of whether the pin is defined as an input or output. This allows detection of output conflicts at the pin by comparing the written data with the data on the pin.

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

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

37.2.2.1 Port A Data Register (PDATA)

	0							7	8	9	10	11	12	13	14	15
Field	—								D8	D9	D10	D11	D12	D13	D14	D15
Reset	—															
R/W	R/W															
Addr	0x10D10															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	—															
R/W	R/W															
Addr	0x10D12															

Figure 37-5. Port A Data Registers (PDATA)

37.2.2.2 Port B Data Register (PDATB)

	0	15
Field	—	
Reset	—	
R/W	R/W	
Addr	0x10D30	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	—															
R/W	R/W															
Addr	0x10D32															

Figure 37-6. Port B Data Registers (PDATB)

37.2.2.3 Port C Data Register (PDATC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	D0	D1	—		D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	—															
R/W	R/W															
Addr	0x10D50															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	—	
Reset	—															
R/W	R/W															
Addr	0x10D52															

Figure 37-7. Port C Data Registers (PDATC)

37.2.2.4 Port D Data Register (PDATD)

	0	6	7	8	13	14	15
Field	—		D7	—		D14	D15
Reset	—						
R/W	R/W						
Addr	0x10D70						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	—		D29	D30	D31
Reset	—														
R/W	R/W														
Addr	0x10D72														

Figure 37-8. Port D Data Registers (PDATD)

37.2.3 Port Data Direction Registers (PDIR*x*)

The port data direction registers (PDIR*x*), shown in Figure 37-9, Figure 37-10, Figure 37-11, and Figure 37-12, are cleared at system reset.

37.2.3.1 Port A Data Direction Register (PDIRA)

	0	7	8	9	10	11	12	13	14	15
Field	—		DR8	DR9	DR10	DR11	DR12	DR13	DR14	DR15
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	0x10D00									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	DR30	DR31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D02															

Figure 37-9. Port A Data Direction Register (PDIRA)

Table 37-5 describes PDIRA fields.

Table 37-5. PDIRA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared
8–31	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

37.2.3.2 Port B Data Direction Register (PDIRB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10D20	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	DR30	DR31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D22															

Figure 37-10. Port B Data Direction Register (PDIRB)

Table 37-6 describes PDIRB fields.

Table 37-6. PDIRB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18–31	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

37.2.3.3 Port C Data Direction Register (PDIRC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DR0	DR1	—		DR4	DR5	DR6	DR7	DR8	DR9	DR10	DR11	DR12	DR13	DR14	DR15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D40															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D42															

Figure 37-11. Port C Data Direction Register (PDIRC)

Table 37-7 describes PDIRC fields.

Table 37-7. PDIRC Field Descriptions

Bits	Name	Description
0–1	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.
2–3	—	Reserved, should be cleared
4–29	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.
30–31	—	Reserved, should be cleared

37.2.3.4 Port D Data Direction Register (PDIRD)

	0	6	7	8	13	14	15
Field	—		DR7	—		DR14	DR15
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10D60						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	—		DR29	DR30	DR31
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x10D62														

Figure 37-12. Port D Data Direction Register (PDIRD)

Table 37-8 describes PDIRD fields.

Table 37-8. PDIRD Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared
7	DR*7*	Direction. Indicates whether PD[7] is used as an input or an output. 0 PD[7] is an input or is bidirectional. 1 PD[7] is an output.
8–13	—	Reserved, should be cleared
14–25	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.
26–28	—	Reserved, should be cleared
29–31	DR*x*	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

37.2.4 Port Pin Assignment Registers (PPAR*x*)

The port pin assignment registers (PPAR*x*) are cleared at system reset.

37.2.4.1 Port A Pin Assignment Registers (PPARA)

	0	7	8	9	10	11	12	13	14	15
Field	—		DD8	DD9	DD10	DD11	DD12	DD13	DD14	DD15
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	0x10D04									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22[1]	DD23[1]	DD24	DD25	DD26	DD27	DD28	DD29	DD30	DD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D06															

[1] MPC8272 only. Reserved on MPC8248.

Figure 37-13. Port A Pin Assignment Register (PPARA)

Table 37-9 describes PPARA fields.

Table 37-9. PPARA Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–21	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRA.
22–23	DD*x*	MPC8272: Dedicated enable. Refer to description above.
	—	MPC8248: Reserved, should be cleared. **Note:** PA[22–23] can be used only as general purpose.
24–31	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRA.

37.2.4.2 Port B Pin Assignment Registers (PPARB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10D24	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	DD30	DD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D26															

Figure 37-14. Port B Pin Assignment Register (PPARB)

Table 37-10 describes PPARB fields.

Table 37-10. PPARB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18–31	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRB.

37.2.4.3 Port C Pin Assignment Registers (PPARC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DD0	DD1	—		DD4	DD5	DD6	DD7	DD8	DD9	DD10	DD11	DD12	DD13	DD14	DD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D44															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D46															

Figure 37-15. Port C Pin Assignment Register (PPARC)

Table 37-11 describes PPARC fields.

Table 37-11. PPARC Field Descriptions

Bits	Name	Description
0–1	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRC.
2–3	—	Reserved, should be cleared
4–29	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRC.
30–31	—	Reserved, should be cleared

37.2.4.4 Port D Pin Assignment Registers (PPARD)

	0	6	7	8	13	14	15
Field	—		DD7	—		DD14	DD15
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10D64						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	—		DD29	DD30	DD31
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x10D66														

Figure 37-16. Port D Pin Assignment Register (PPARD)

Table 37-12 describes PPARD fields.

Table 37-12. PPARD Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared
7	DD*7*	Dedicated enable. Indicates whether PD[7] is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of PD[7] are not used. 1 Dedicated peripheral function. PD[7] is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as PDIRD[7].
8–13	—	Reserved, should be cleared
14–25	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRD.

Table 37-12. PPARD Field Descriptions (continued)

Bits	Name	Description
26–28	—	Reserved, should be cleared
29–31	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those in the PDIRD.

37.2.5 Port Special Options Registers (PSOR*x*)

PSOR*x* bits are effective only if the corresponding PPAR*x*[DD*x*] = 1 (a dedicated peripheral function).

NOTE

> If the corresponding PPAR*x*[DD*x*] = 1 (configured as a general-purpose pin) before programming a PSOR*x* or PDIR*x* bit, a pin might function for a short period as an unwanted dedicated function and cause unknown behavior.

Figure 37-17 shows the port special options registers (PSOR*x*).

37.2.5.1 Port A Special Options Register (PSORA)

	0–7	8	9	10	11	12	13	14	15
Field	—	SO8	SO9	SO10	SO11	SO12	SO13	SO14	SO15
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x10D08								

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22[1]	SO23[1]	SO24	SO25	SO26	SO27	SO28	SO29	SO30	SO31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D0A															

[1] MPC8272 only. Reserved on MPC8248.

Figure 37-17. Port A Special Options Register (PSORA)

Table 37-13 describes PSORA fields.

Table 37-13. PSOR*x* Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–21	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPARA[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
22–23	SO*x*	MPC8272: Special-option. See description above.
	—	MPC8248: Reserved, should be cleared. **Note:** PA[22–23] can be used only as general purpose.
24–31	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPARA[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

37.2.5.2 Port B Special Options Registers (PSORB)

	0	15
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x10D28	

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—		SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	SO30	SO31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D2A															

Figure 37-18. Port B Special Options Registers (PSORB)

Table 37-14 describes PSORB fields.

Table 37-14. PSORB Field Descriptions

Bits	Name	Description
0–17	—	Reserved, should be cleared
18–31	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPARB[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

37.2.5.3 Port C Special Options Registers (PSORC)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SO0	SO1	—		SO4	SO5	SO6	SO7	SO8	SO9	SO10	SO11	SO12	SO13	SO14	SO15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D48															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	—	
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x10D4A															

Figure 37-19. Port C Special Options Registers (PSORC)

Table 37-15 describes PSORC fields.

Table 37-15. PSORC Field Descriptions

Bits	Name	Description
0–1	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPAR*C*[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
2–3	—	Reserved, should be cleared
4–29	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPAR*C*[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
30–31	—	Reserved, should be cleared

37.2.5.4 Port D Special Options Registers (PSORD)

	0	6	7	8	13	14	15
Field	—		SO7	–		SO14	SO15
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x10D68						

	16	17	18	19	20	21	22	23	24	25	26	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	–		SO29	SO30	SO31
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x10D6A														

Figure 37-20. Special Options Registers (PSORD)

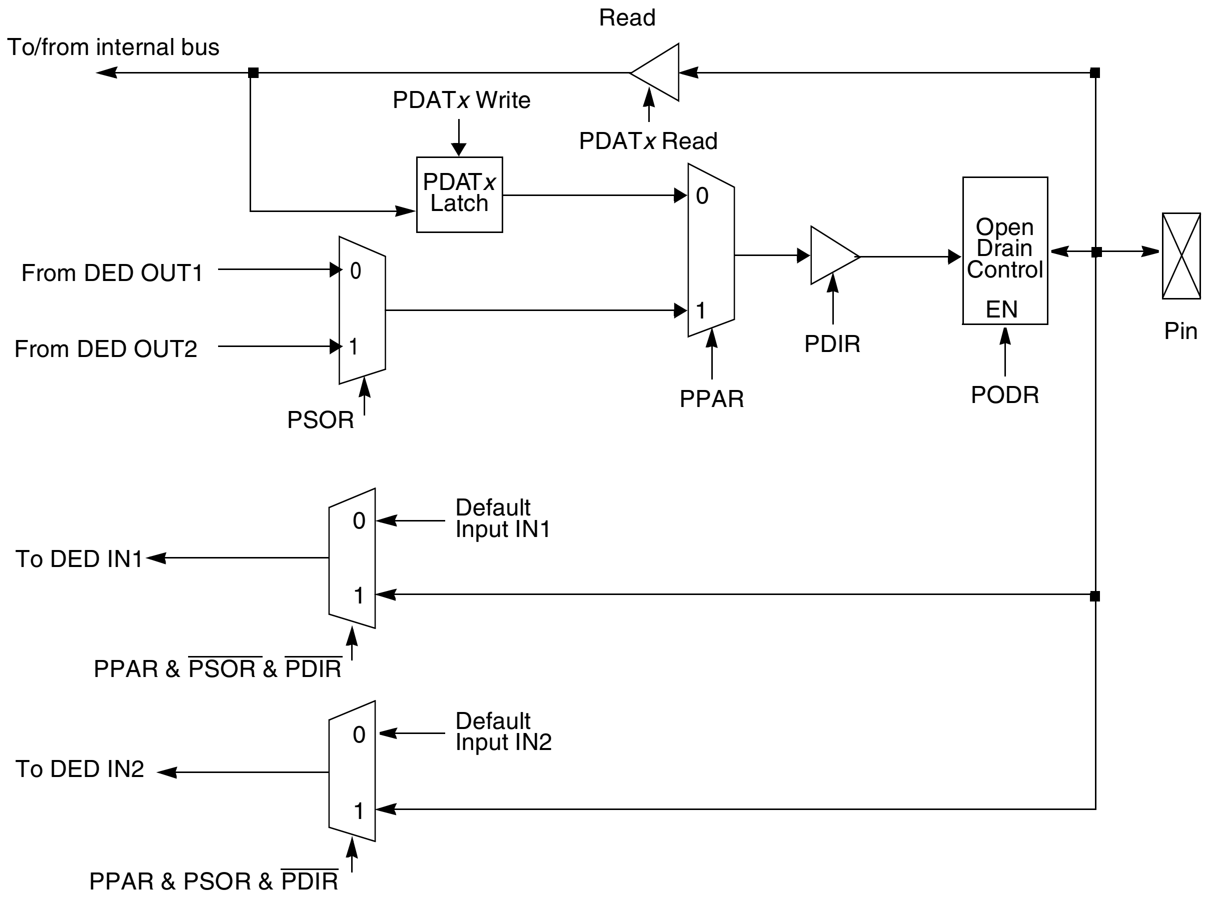
Table 37-13 describes PSORD fields.

Table 37-16. PSORD Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared
7	SO*7*	Special-option. Determines whether PD[7] when configured as a dedicated function (PPARD[DD*7*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
8–13	—	Reserved, should be cleared.
14–25	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPARD[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.
26–28	—	Reserved, should be cleared.
29–31	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPARD[DD*x*] = 1) uses option 1 or option 2. Options are described in Section 37.5, “Ports Tables.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

37.3 Port Block Diagram

Figure 37-21 shows the functional block diagram.



Register Name	0	1	Description
PPAR*x*	General purpose	Dedicated	Port pin assignment
PSOR*x*	Dedicated 1	Dedicated 2	Special operation
PDIR*x*	Input	Output	Direction[1]
PODR*x*	Regular	Open drain	
PDAT*x*	0	1	Data

[1] Bidirectional signals must be programmed as inputs (PDIR = 0).

Figure 37-21. Port Functional Operation

37.4 Port Pins Functions

Each pin can operate as a general-purpose I/O pin or as a dedicated input or output pin.

37.4.1 General-Purpose I/O Pins

Each of the port pins is independently configured as a general-purpose I/O pin if the corresponding port pin assignment register (PPAR) bit is cleared. Each pin is configured as a dedicated on-chip peripheral pin if the corresponding PPAR bit is set.When the port pin is configured as a general-purpose I/O pin, the signal direction for that pin is determined by the corresponding control bit in the port data direction register (PDIR). The port I/O pin is configured as an input if the corresponding PDIR bit is cleared; it is configured as an output if the corresponding PDIR bit is set. All PPAR and PDIR bits are cleared on total system reset, configuring all port pins as general-purpose input pins.

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

37.4.2 Dedicated Pins

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

NOTE

> Some output functions can be output on two different pins. For example, the output for BRG5 can come out on both PC13 and PC23. The user can freely configure such functions to be output on two pins at once. However, there is typically no advantage in doing so unless there is a large fanout where it is advantageous to share the load between two pins.
>
> Many input functions can also come from two different pins; see Section 37.5, “Ports Tables.”

37.5 Ports Tables

Table 37-17 through Table 37-20 describe the ports functionality according to the configuration of the port registers (PPAR*x*, PSOR*x*, and PDIR*x*). Each pin can function as a general-purpose I/O, one of two dedicated outputs, or one of two dedicated inputs.

As shown in Figure 37-22, some input functions can come from two different pins for flexibility. Secondary option programming is relevant only if primary option is programmed to the default value.

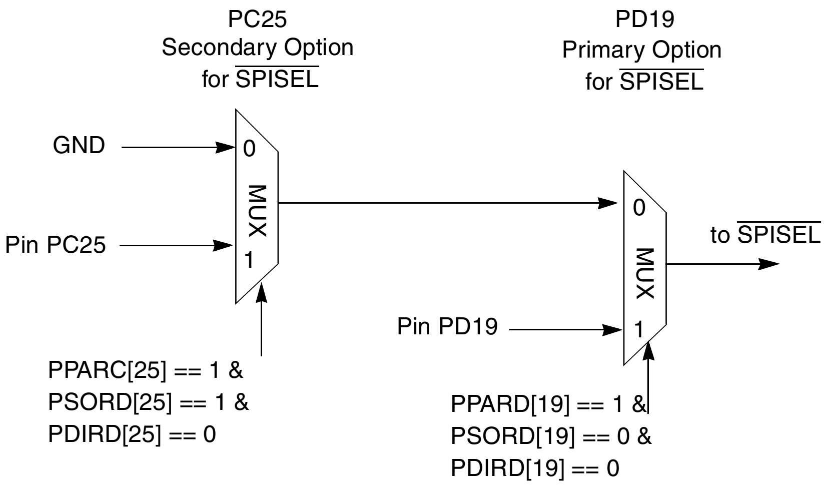


Figure 37-22. Primary and Secondary Option Programming

In the tables below, the default value for a primary option is simply a reference to the secondary option. In the secondary option, the programming is relevant only if the primary option is not used for the function.

Table 37-17 shows the port A pin assignments.

Table 37-17. Port A—Dedicated Pin Assignment (PPARA = 1)

Pin	Pin Function					
	PSORA = 0			**PSORA = 1**		
	PDIRA = 1 (Output)	**PDIRA = 0 (Input)**	**Default Input**	**PDIRA = 1 (Output)**	**PDIRA = 0 (Input, or Inout if Specified)**	**Default Input**
PA31	**FCC1: TxEnb** UTOPIA master[1]	**FCC1: TxEnb** UTOPIA slave[1]	GND		**FCC1: COL** MII	GND
PA30	**FCC1: TxClav** UTOPIA slave[1]	**FCC1: TxClav** UTOPIA master[1] **FCC1: TxClav0** MPHY, master, direct polling	GND	**FCC1: $\overline{\text{RTS}}$**	**FCC1: CRS** MII	GND
PA29	**FCC1: TxSOC** UTOPIA[1]			**FCC1: TX_ER** MII		
PA28	**FCC1: RxEnb** UTOPIA master[1]	**FCC1: RxEnb** UTOPIA slave[1]	GND	**FCC1: TX_EN** MII/RMII		
PA27		**FCC1: RxSOC** UTOPIA[1]	GND		**FCC1: RX_DV** MII **FCC1: CRS_DV** RMII	GND
PA26	**FCC1: RxClav** UTOPIA slave[1]	**FCC1: RxClav** UTOPIA master[1] **FCC1: RxClav0** MPHY, master, direct polling	GND		**FCC1: RX_ER** MII/RMII	GND

Table 37-17. Port A—Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	**Pin Function**					
	PSORA = 0			**PSORA = 1**		
	PDIRA = 1 (Output)	**PDIRA = 0 (Input)**	**Defaul t Input**	**PDIRA = 1 (Output)**	**PDIRA = 0 (Input, or Inout if Specified)**	**Defaul t Input**
PA25	**FCC1: TxD[0]** UTOPIA 8[1]			MSNUM[0][2]		
PA24	**FCC1: TxD[1]** UTOPIA 8[1]			MSNUM[1][2]		
PA23	**FCC1: TxD[2]** UTOPIA 8[1]					
PA22	**FCC1: TxD[3]** UTOPIA 8[1]					
PA21	**FCC1: TxD[4]** UTOPIA 8[1] **FCC1: TxD[3]** MII/HDLC nibble					
PA20	**FCC1: TxD[5]** UTOPIA 8[1] **FCC1: TxD[2]** MII/HDLC nibble					
PA19	**FCC1: TxD[6]** UTOPIA 8[1] **FCC1: TxD[1]** MII/HDLC nibble **FCC1: TxD[1]** **RMII dibit**					
PA18	**FCC1: TxD[7]** UTOPIA 8[1] **FCC1: TxD[0]** MII/HDLC nibble **FCC1: TxD[0]** RMII dibit **FCC1: TxD** HDLC/transp					
PA17		**FCC1: RxD[7]** UTOPIA 8[1] **FCC1: RxD[0]** MII/HDLC nibble **FCC1: RxD[0]** **RMII dibit** **FCC1: RxD[0]** HDLC/transp.	GND			

Table 37-17. Port A—Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or Inout if Specified)	Default Input
PA16		**FCC1: RxD[6]** UTOPIA 8[1] **FCC1: RxD[1]** MII/HDLC nibble **FCC1: RxD[1]** **RMII dibit**	GND			
PA15		**FCC1: RxD[5]** UTOPIA 8[1] **FCC1: RxD[2]** MII/HDLC nibble	GND			
PA14		**FCC1: RxD[4]** UTOPIA 8[1] **FCC1: RxD[3]** MII/HDLC nibble	GND			
PA13		**FCC1: RxD[3]** UTOPIA 8[1]	GND	MSNUM[2][2]		
PA12		**FCC1: RxD[2]** UTOPIA 8[1]	GND	MSNUM[3][2]		
PA11		**FCC1: RxD[1]** UTOPIA 8[1]	GND	MSNUM[4][2]		
PA10		**FCC1: RxD[0]** UTOPIA 8[1]	GND	MSNUM[5][2]		
PA9	**SMC2: SMTXD**					
PA8		**SMC2: SMRXD** (primary option)	GND			

[1] MPC8272 only.

[2] MSNUM[0–4] is the sub-block code of the peripheral controller using SDMA; MSNUM[5] indicates which section, transmit or receive, is active during the transfer. See Section 18.2.4, "SDMA Transfer Error MSNUM Register (PDTEM)."

Table 37-18 shows the port B pin assignments.

Table 37-18. Port B Dedicated Pin Assignment (PPARB = 1)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or Inout if Specified)	Default Input
PB31	**FCC2: TX_ER** MII					
PB30		**FCC2: RX_DV** MII **FCC2: CRS_DV** RMII	GND			
PB29				**FCC2: TX_EN** MII/RMII		
PB28	**FCC2: $\overline{\text{RTS}}$**	**FCC2: RX_ER** MII/RMII	GND	**SCC1: TXD**		
PB27		**FCC2: COL** MII	GND		**TDM_B2: L1TXD** Inout	GND
PB26		**FCC2: CRS** MII	GND		**TDM_B2: L1RXD** Inout	GND
PB25	**FCC2: TxD[3]** MII/HDLC nibble				**TDM_B2: L1TSYNC/GRANT**	GND
PB24	**FCC2: TxD[2]** MII/HDLC nibble				**TDM_B2: L1RSYNC**	GND
PB23	**FCC2: TxD[1]** MII/HDLC nibble **FCC2: TxD[1]** RMII dibit					
PB22	**FCC2: TxD[0]** MII/HDLC nibble **FCC2: TxD[0]** RMII dibit **FCC2: TxD** HDLC/transp. serial					
PB21		**FCC2: RxD[0]** MII/HDLC nibble **FCC2: RxD[0]** RMII dibit **FCC2: RxD** HDLC/transp. serial	GND			
PB20		**FCC2: RxD[1]** MII/HDLC nibble **FCC2: RxD[1]** RMII dibit	GND			

Table 37-18. Port B Dedicated Pin Assignment (PPARB = 1) (continued)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or Inout if Specified)	Default Input
PB19		**FCC2: RxD[2]** MII/HDLC nibble	GND	**TDM_B2: $\overline{\text{L1RQ}}$**		
PB18		**FCC2: RxD[3]** MII/HDLC nibble	GND	**TDM_B2: L1CLKO**		

Table 37-19 shows the port C pin assignments.

Table 37-19. Port C Dedicated Pin Assignment (PPARC = 1)

Pin	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC29	**BRG2: BRGO**	CLK3/TIN2	CLK7		**SCC1: $\overline{\text{CTS}}$**[1] **SCC1: CLSN**[1] Ethernet (secondary option)	GND
PC28	**Timer2: $\overline{\text{TOUT}}$**	CLK4/TIN1	CLK8		**SPI: SPICLK**[1] Inout (secondary option)	GND
PC27	**Timer1: $\overline{\text{TOUT}}$**	CLK5	GND	**BRG3: BRGO**	**FCC1: RxPrty** UTOPIA[2] (secondary option)	GND
PC26	**Timer3: $\overline{\text{TOUT}}$**	CLK6	GND		TMCLK real-time counter	BRGO1
PC25	**IDMA2: $\overline{\text{DACK}}$**	CLK7/TIN4	GND	**BRG4: BRGO**	**SPI: $\overline{\text{SPISEL}}$**[1] (secondary option)	V_{DD}
PC24	**BRG1: BRGO**	CLK8/TIN3	GND	**Timer4: $\overline{\text{TOUT}}$**	**IDMA2: DREQ**[3]	GND
PC23	**BRG5: BRGO**	CLK9	CLK13	**IDMA3: $\overline{\text{DACK}}$**	**SCC1: $\overline{\text{CD}}$** **SCC1: RENA** Ethernet (secondary option)	GND
PC22	**FCC1: TxPrty** UTOPIA[2]	CLK10	CLK14		**IDMA3: $\overline{\text{DONE}}$** Inout	V_{DD}
PC21	**BRG6: BRGO**	CLK11	CLK15		**CP_INT**	GND
PC20	**USB: $\overline{\text{OE}}$**	CLK12	CLK16			

Table 37-19. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

Pin	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC19	**BRG7: BRGO**	CLK13	GND		**timer1/2: $\overline{\text{TGATE1}}$**	GND
PC18		CLK14	GND		**timer3/4: $\overline{\text{TGATE2}}$**	GND
PC17	**BRG8: BRGO**	CLK15	GND		**IDMA2: $\overline{\text{DONE}}$** Inout	V_{DD}
PC16		CLK16	GND			
PC15		**SCC1: $\overline{\text{CTS}}$** **SCC1: CLSN** Ethernet (primary option)	by PC29	**FCC1: TxAddr[0]** MPHY, master	**FCC1: TxAddr[0]** MPHY, slave	GND
PC14		**SCC1: $\overline{\text{CD}}$** **SCC1: RENA** Ethernet (primary option)	by PC23	**FCC1: RxAddr[0]** MPHY, master	**FCC1: RxAddr[0]** MPHY, slave	GND
PC13	**BRG5: BRGO**			**FCC1: TxAddr[1]** MPHY, master	**FCC1: TxAddr[1]** MPHY, slave	GND
PC12			V_{DD}	**FCC1: RxAddr[1]** MPHY, master	**FCC1: RxAddr[1]** MPHY, slave	GND
PC11		**SCC3: $\overline{\text{CTS}}$** **SCC3: CLSN** Ethernet **USB: RP** (primary option)	by PC8			
PC10		**SCC3: $\overline{\text{CD}}$** **SCC3: RENA** Ethernet **USB: RN**	GND			
PC9		**SCC4: $\overline{\text{CTS}}$** **SCC4: CLSN /** Ethernet	GND		**TDM_A2:** **L1TSYNC/GRANT**	GND
PC8	**SCC1: $\overline{\text{RTS}}$** **SCC1: TENA**	**SCC4: $\overline{\text{CD}}$** **SCC4: RENA /** Ethernet	GND	**SI2: L1ST1** Strobe	**SCC3: $\overline{\text{CTS}}$**[1] **SCC3: CLSN**[1] (secondary option)	GND
PC7		**FCC1: $\overline{\text{CTS}}$**	GND	**FCC1: TxAddr[2]** MPHY master, multiplexed: polling	**FCC1: TxAddr[2]** MPHY, slave, multiplexed polling **FCC1: TxClav1** MPHY, master, direct polling	GND

Table 37-19. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

Pin	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC6	**SI2: L1ST2** Strobe	**FCC1: $\overline{CD}$**	GND	**FCC1: RxAddr[2]** MPHY, master, multiplexed polling	**FCC1: RxAddr[2]** MPHY, slave, multiplexed polling) **FCC1: RxClav1** MPHY, master, direct polling	GND
PC5	**SMC1: SMTXD**			**SI2: L1ST3** Strobe	**FCC2: $\overline{CTS}$**	GND
PC4		**SMC1: SMRXD**	GND	**SI2: L1ST4** Strobe	**FCC2: $\overline{CD}$**	GND
PC1	**BRG6: BRGO**		by PC24	**TDM_A2: $\overline{L1RQ}$**		
PC0	**BRG7: BRGO**	**IDMA3: DREQ**[3]	GND	**TDM_A2: L1CLKO**	**SMC1: SMSYN**	GND

[1] Available only when the primary option for this function is not used.

[2] MPC8272 only.

[3] Do not program the IDMA*x*-DREQ pins to assert external requests to the IDMA, unless the IDMA is used. Otherwise, erratic operation occurs.

Table 37-20 shows the port D pin assignments.

Table 37-20. Port D Dedicated Pin Assignment (PPARD = 1)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or Inout if Specified)	Default Input
PD31		**SCC1: RXD**	GND			
PD30				**SCC1: TXD**		
PD29	**SCC1: $\overline{RTS}$** **SCC1: TENA** Ethernet			**FCC1: RxAddr[3]** MPHY, master, multiplexed polling	**FCC1: RxAddr[3]** MPHY, slave, multiplexed polling **FCC1: RxClav2** MPHY, master, direct polling	GND
PD25		**SCC3: RX/** **USB: Rxd**	GND			

Table 37-20. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or Inout if Specified)	Default Input
PD24	**SCC3: TXD/** **USB: TN**					
PD23	**SCC3: $\overline{\text{RTS}}$** **SCC3: TENA** Ethernet **USB: TP**					
PD22		**SCC4: RXD**	GND		**TDM_A2: L1TXD** Inout	GND
PD21	**SCC4: TXD**				**TDM_A2: L1RXD** Inout	GND
PD20	**SCC4: $\overline{\text{RTS}}$** **SCC4: TENA** Ethernet				**TDM_A2: L1RSYNC**	GND
PD19	**FCC1: TxAddr[4]** MPHY, master, multiplexed polling	**FCC1: TxAddr[4]** MPHY, slave, multiplexed polling **FCC1: TxClav3** MPHY, master, direct polling	GND	**BRG1: BRGO**	**SPI: $\overline{\text{SPISEL}}$** (primary option)	PC25
PD18	**FCC1: RxAddr[4]** MPHY, master, multiplexed polling	**FCC1: RxAddr[4]** MPHY, slave, multiplexed polling **FCC1: RxClav3** MPHY, master, direct polling	GND		**SPI: SPICLK** Inout (primary option)	PC28
PD17	**BRG2: BRGO**	**FCC1: RxPrty** UTOPIA[1] (primary option)	PC27		**SPI: SPIMOSI** Inout	V_{DD}
PD16	**FCC1: TxPrty** UTOPIA[1]				**SPI: SPIMISO** Inout	SPIMOSI
PD15					**I2C: I2CSDA** Inout	V_{DD}
PD14					**I2C: I2CSCL** Inout	GND
PD7		**SMC2: SMSYN**	GND	**FCC1: TxAddr[3]** MPHY, master, multiplexed polling	**FCC1: TxAddr[3]** MPHY, slave, multiplexed polling **FCC1: TxClav2** MPHY, master, direct polling	GND

[1] MPC8272 only.

37.6 Interrupts from Port C

The port C lines associated with $\overline{\text{CD}x}$ and $\overline{\text{CTS}x}$ have a mode of operation where the pin can be internally connected to the SCC/FCC but can also generate interrupts. Port C still detects changes on the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins and asserts the corresponding interrupt request, but the SCC/FCC simultaneously uses $\overline{\text{CTS}}$ and/or $\overline{\text{CD}}$ to automatically control operation. This lets the user fully implement protocols V.24, X.21, and X.21 bis (with the assistance of other general-purpose I/O lines).

Follow these steps to configure a port C pin as a $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ pin that connects to the SCC/FCC and generates interrupts:

1. Write the corresponding PPARC bit with a 1 and PSORC bit with 0.
2. Write the corresponding PDIRC bit with a zero.
3. Set the SIEXR bit (in the interrupt controller) to determine which edges cause interrupts.
4. Write the corresponding SIMR (mask register) bit with a 1 to allow interrupts to be generated to the core.
5. The pin value can be read at any time using PDATC.

NOTE

> After connecting $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ to the SCC/FCC, the user must also choose the normal operation mode in GSMR[DIAG] to enable and disable SCC/FCC transmission and reception with these pins.

Part V
Integrated Security Engine

Intended Audience

This part is intended for system designers who need to utilize the encryption functionality on the MPC8272. It assumes a basic understanding of the PowerPC exception model and the MPC8272 interrupt structure.

Contents

This part describes behavior of the MPC8272 integrated security engine (SEC).

It contains the following chapter:

- Chapter 38, “Security Engine (SEC),” describes the SEC, which is designed to off load computational intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the MPC8272.

Suggested Reading

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

MPC82xx Documentation

Supporting documentation for the MPC8272 can be accessed through the world-wide web at www.freescale.com. This documentation includes technical specifications, reference materials, and detailed applications notes.

Architecture Documentation

Documentation is available in the following document:

- Programming environments manuals—These books provide information about resources defined by the PowerPC architecture that are common to processors that implement the PowerPC architecture. There are two versions, one that describes the functionality of the combined 32- and 64-bit architecture models and one that describes only the 32-bit model.
  - — *Programming Environments for 32-Bit Implementations of the PowerPC Architecture*, REV 3(Freescale order #: MPCFPE32B/AD)

For a current list of documentation, refer to http://www.freescale.com.

Conventions

This document uses the following notational conventions:

Bold	Bold entries in figures and tables showing registers and parameter RAM should be initialized by the user.
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.
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 or acronyms for registers or buffer descriptors are shown in uppercase text. Specific bits, fields, or numerical ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In certain contexts, such as in a signal encoding or a bit field, indicates a don’t care.
n	Indicates an undefined numerical value
¬	NOT logical operator
&	AND logical operator
|	OR logical operator

Chapter 38
Security Engine (SEC)

This chapter describes the functionality of the MPC8272's integrated security engine (SEC). It addresses the following topics:

The SEC is designed to off load computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the MPC8272. It is optimized to process all the algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i.

The security engine's execution units (EUs) and primary features include the following:

- PKEU—Public key execution unit that supports the following:
  - RSA and Diffie-Hellman
    - Programmable field size up to 2048 bits
  - Elliptic curve cryptography
    - F_2m and F(p) modes
    - Programmable field size up to 511 bits
- DEU—Data encryption standard execution unit
  - DES, 3DES
  - Two key (K1, K2, K1) or three Key (K1, K2, K3)
  - ECB and CBC modes for both DES and 3DES
- AESU—Advanced encryption standard unit
  - Implements the Rinjdael symmetric key cipher
  - ECB, CBC, CCM, and counter modes
  - 128-, 192-, 256-bit key lengths
- AFEU—ARC four execution unit
  - Implements a stream cipher compatible with the RC4 algorithm
  - 40- to 128-bit programmable key

- MDEU—Message digest execution unit
  - — SHA with 160- or 256-bit message digest
  - — MD5 with 128-bit message digest
  - — HMAC with either algorithm
- RNG—1 Random number generator
- Master/slave logic, with DMA
  - — 32-bit address/64-bit data
  - — Up to 100-MHz operation
- Four crypto-channels, each supporting multi-command descriptor chains
  - — Static and/or dynamic assignment of execution units through an integrated controller
  - — Buffer size of 512 bytes per execution unit, with flow control for large data sizes

38.1 SEC Architecture Overview

The SEC can act as a master on the internal bus. This allows the SEC to off load the data movement bottleneck normally associated with slave-only cores. The host processor accesses the SEC through its device drivers using system memory for data storage. The SEC resides in the peripheral memory map of the processor; therefore, when an application requires cryptographic functions, it simply creates descriptors for the SEC 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.

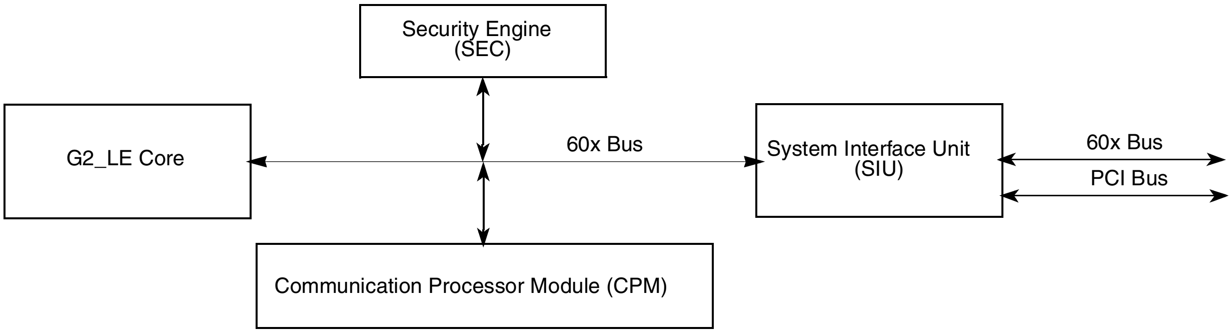

Security Engine (SEC)
G2_LE Core
60x Bus
System Interface Unit (SIU)
60x Bus
PCI Bus
Communication Processor Module (CPM)

Figure 38-1. SEC Connected to 60x Bus

A block diagram of the SEC internal architecture is shown in Figure 38-2. The bus interface module is designed to transfer 64-bit words between the bus and any register inside the SEC.

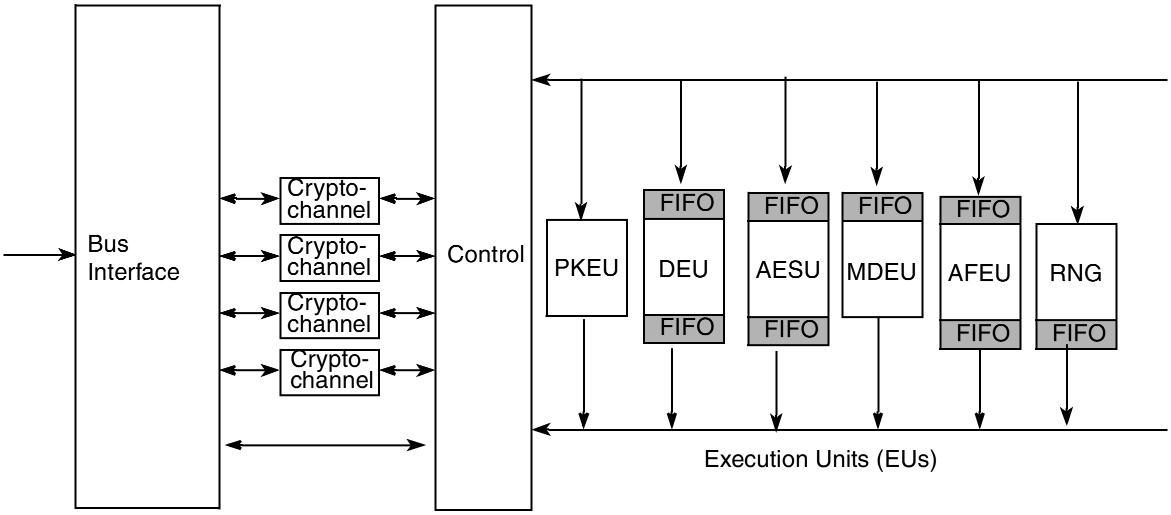


Figure 38-2. SEC Functional Modules

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 per the programmable priority scheme. As data is processed, it is written to the individual execution units output buffer and then back to system memory through the master/slave interface module.

38.1.1 Data Packet Descriptors

As a crypto-acceleration block, the SEC controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through data packet descriptors, some of which have been defined as multifunction to facilitate IPSec applications. A data packet descriptor is diagrammed in Table 38-1.

Table 38-1. 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

Table 38-1. Example Data Packet Descriptor (continued)

Field Name	Value/Type	Description
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
PTR_NEXT	Pointer	Pointer to next data packet descriptor

Each data packet descriptor contains the following:

- Header—Describes the required services and encodes information that indicates which EUs to use and which modes to set
- Seven data length/data pointer pairs—Indicates the number of contiguous bytes of data to be transferred. The data pointer indicates the starting address of the data, key, or context in system memory.
- Next descriptor pointer

A data packet descriptor ends with a pointer to the next data packet descriptor. Upon completion of the current descriptor, this field is checked and, if non-zero, the channel is instructed to request a burst read of the next descriptor.

Processing of the next descriptor (and whether or not a done signal is generated) is determined by the programming of crypto-channel’s configuration register. Two modes of operation are supported:

- Signal done at end of descriptor
- Signal done at end of descriptor chain

The crypto-channel can signal done through an interrupt or by a write-back of the descriptor header after processing a data packet descriptor. The value written back is identical to that of the header, with the exception that a DONE field is set.

Occasionally, a descriptor field may not be applicable to the requested service. For example, if using DES in ECB mode, the contents of the IV field do not affect the result of the DES computation. Therefore, when processing data packet descriptors, the crypto-channel skips any pointer that has an associated length of zero.

For more information, refer to Section 38.4, “Data Packet Descriptors."

38.1.2 Execution Units (EUs)

Execution unit (EU) is the generic term for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPSec, IKE, SSL/TLS, iSCSI, SRTP, and 802.11i 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-4 compatible stream cipher symmetric key cryptography
- AESU for performing the advanced encryption standard algorithm
- MDEU for performing security hashing using MD-5, SHA-1, or SHA-256
- RNG for random number generation

Each EU is described in detail in Section 38.5, "Execution Units", on page 27.

38.1.2.1 Public Key Execution Unit (PKEU)

The PKEU is capable of performing many advanced mathematical functions to support both RSA and ECC public key cryptographic algorithms. ECC is supported in both F(2)m (polynomial-basis) and F(p) modes. This EU supports all levels of functions to assist the host microprocessor to perform its desired cryptographic function. 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 38.5.1, “Public Key Execution Units (PKEUs)."

38.1.2.1.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 160 bits and 512 bits in programmable increments of 8, with each programmable value *i* supporting all actual field sizes from 8*i* – 7 to 8*i*. The result is hardware supporting a wide range of cryptographic security. Larger field / modulus sizes result in greater security but lower performance; processing time is determined by field or modulus size. For example, a field size of 160 is roughly equivalent to the security provided by 1024-bit RSA. A field size set to 208 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations: 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.

38.1.2.1.2 Modular Exponentiation Operations

The PKEU is also capable of performing ordinary integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the SEC’s PKEU include the following:

- R 2 mod N
- A’ E mod N
- $(A \times B)\ R^{-1}$ mod N
- $(A \times B)\ R^{-2}$ mod N
- $(A + B)$ mod N

- (A – B) mod N

With 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 129 bits. The PKEU uses the Montgomery modular multiplication algorithm to perform core functions. The addition and subtraction functions exist to help support known methods of the Chinese remainder theorem (CRT) for efficient exponentiation.

38.1.2.2 Data Encryption Standard Execution Unit (DEU)

The DES execution unit (DEU) performs bulk data encryption/decryption, in compliance with the data encryption standard algorithm (ANSI x3.92). The DEU can also compute 3DES, an extension of the DES algorithm in which each 64-bit input block is processed three times. The SEC supports two-key (K1 = K3) and three-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 IV operation: electronic code book (ECB) and cipher clock chaining (CBC).

For more information, refer to Section 38.5.2, “Data Encryption Standard Execution Units (DEUs)."

38.1.2.3 Arc Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning a byte of plain text is encrypted with a key to produce a byte of ciphertext. The key is variable length and the AFEU supports key lengths from 40 to 128 bits (in byte increments), providing a wide range of security strengths. RC4 is a symmetric algorithm, meaning each of the two communicating parties share the same key.

For more information, refer to Section 38.5.3, “ARC Four Execution Units (AFEUs)."

38.1.2.4 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the advanced encryption standard algorithm Rinjdael. 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 128-bit input. The AESU operates in ECB, CBC, and counter modes.

For more information, refer to Section 38.5.6, “Advanced Encryption Standard Execution Units (AESUs)."

38.1.2.5 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using one of 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 potential, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
- SHA-256 is a 256-bit hash function that provides 256 bits of security against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

For more information, refer to Section 38.5.4, “Message Digest Execution Units (MDEUs)."

38.1.2.6 Random Number Generator (RNG)

The RNG is a digital integrated circuit capable of generating 32-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 38.5.5, “Random Number Generator (RNG)."

38.1.3 Crypto-Channels

The SEC includes four crypto-channels that manage data and EU function. Each crypto-channel consists of the following:

- Control registers containing information about the transaction in process
- A status register containing an indication of the last unfulfilled bus request
- A pointer register indicating the location of a new descriptor to fetch
- Buffer memory used to store the active data packet descriptor

Crypto-channels analyze the data packet descriptor header and request the first required cryptographic service from the controller. The controller implements a programmable prioritization scheme that allows the user to dictate the order in which the four crypto-channels are serviced. After the controller grants access to the required EU, the crypto-channel and the controller perform the following steps:

1. Set the appropriate mode bits available in the EU for the required service.

2. Fetch context and other parameters as indicated in the data packet descriptor buffer and use these to program the EU.
3. Fetch data as indicated and place in either the EU input FIFO or the EU itself (as appropriate).
4. Wait for EU to complete processing.
5. Upon completion, unload results and context and write them to external memory as indicated by the data packet descriptor buffer.
6. If multiple services requested, go back to step 2.
7. Reset the appropriate EU if it is dynamically assigned. Note that if statically assigned, a EU is reset only upon direct command written to the SEC.
8. Perform descriptor completion notification as appropriate. This notification comes in one of two forms—interrupt or header write-back modification—and can occur at the end of every descriptor, at the end of a descriptor chain, or at the end of specially designated descriptors within a chain.

For more information, refer to Section 38.6, “Crypto-Channels."

38.1.4 SEC Controller

The SEC controller manages on-chip resources, including the individual execution units (EUs), FIFOs, the master/slave interface, and the internal buses that connect all the various modules. The controller receives service requests from the master/slave interface and various crypto-channels, and schedules the required activities. The controller can configure each of the on-chip resources in three modes:

- Host-controlled access—The host is directly responsible for all data movement into and out of the resource.
- Static EU access—The user can reserve a specific execution unit to a specific crypto-channel.
- Dynamic EU access—A crypto-channel can request a particular service from any available execution unit.

38.1.4.1 Host-Controlled Access

All execution units (EU) can be used entirely through register read/write access. It is strongly recommended that read/write access only be performed on a EU that is statically assigned to an idle crypto-channel. Such an assignment is the only method for the host to inform the controller that a particular EU is in use.

38.1.4.2 Static EU Access

The controller can be configured to reserve one or more EUs to a particular crypto-channel. Doing so permits locking the EU to a particular context. When in this mode, the crypto-channel can be used by multiple descriptors representing the same context without unloading and reloading the context at the end of each descriptor. This mode presents considerable performance improvement over dynamic access, but only when the SEC is supporting few (or one) contexts.

38.1.4.3 Dynamic EU Access

Processing begins when a data packet descriptor pointer is written to the next descriptor pointer register of one of the crypto-channels. Prior to fetching the data referred to by the descriptor and based on the services requested by the descriptor header in the descriptor buffer, the controller dynamically reserves usage of an EU to the crypto-channel. If all appropriate EUs are already dynamically reserved by other crypto-channels, the crypto-channel stalls and waits to fetch data until an appropriate EU is available.

If multiple crypto-channels simultaneously request the same EU, the EU is assigned on a weighted priority or round-robin basis. Once the required EU has been reserved, the crypto-channel fetches and loads the appropriate data packets, operates the EU, unloads data to system memory, and releases the EU for use by another crypto-channel. If a crypto-channel attempts to reserve a statically-assigned EU (and no appropriate EUs are available for dynamic assignment), an interrupt is generated and status indicates illegal access. When dynamic assignment is used, each encryption/decryption packet must contain context that is particular to the context being supported.

For more information, refer to Section 38.7, “Controller."

38.1.5 Bus Interface

The master/slave interface manages communication between the SEC’s internal execution units and the MPC8272 internal bus. All on-chip resources are memory mapped, and the slave accesses and initiator writes from the SEC must be addressed on word boundaries. The SEC performs initiator reads on byte boundaries and 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 38.8, “Bus Interface."

38.2 Configuration of Internal Memory Space

The SEC has 128 Kbytes of internal host accessible memory. To access SEC registers, the user should perform the following steps (to be completed once at the setup stage):

1. Program SECMR (SEC mask register) to 0xFFFE_0000.
2. Program SECBR[BA] (SEC base address register). The base address should point to the beginning of an aligned 128-Kbyte free memory space.
3. Enable the bank by setting SECBR[V] = 1.

The SEC base address register is part of the MPC8272’s memory controller. The offset for SEC registers is described in Section 38.3, “Address Map.”

It is recommended to set SECBR[BA] = IMMR + 0x4_0000; this forms a 384-Kbyte contiguous internal memory block.

The address of a given SEC internal register for host access is SECBR[BA] + internal offset.

38.2.1 SEC Address Base Register

SECBR holds the 17 most-significant bits of the base address of the SEC internal memory space. It also enables or disables accesses to that space through bit 31 [Valid]. Figure 38-3 shows SECBR fields, and Table 38-2 describes them.

0	14	15
BA		BA15
0000_0000_0000_0000		
R/W		
IMMR+0x101B4		

16	17	30	31
BA16	—		V
0000_0000_0000_0000			
R/W			
IMMR+0x101B6			

Figure 38-3. SEC Base Register (SECBR)

Table 38-2. SECBR Field Descriptions

Bits	Name	Description
0–14	BA	Base address bits 0–14. User programmable. Recommended to set BA to an offset of 0x4_0000 from IMMR.
15–16	BA[15–16]	Base address bits 15–16. Should be cleared. (BA should be 128-Kbyte aligned),
17–30	—	Reserved. should be cleared.
31	V	Valid bit. Indicates that the contents of SECMR and SECBR are valid. 0 SEC bank is invalid. Access to SEC memory space may cause timeout. 1 SEC bank is valid.

38.2.2 SEC Address Mask Register

SECMR defines the size of the SEC internal memory space. Figure 38-4 shows SECMR fields, and Table 38-3 describes them.

0	15
AM	
0000_0000_0000_0000	
R/W	
IMMR+0x101BC	

16	17	31
AM16	—	
0000_0000_0000_0000		
R/W		
IMMR+0x101BE		

Figure 38-4. SEC Mask Register (SECMR)

Table 38-3. SECMR Field Descriptions

Bits	Name	Description
0–15	AM	Address mask bits 0–15. Should be set to 0xFFFE (128-Kbyte memory space)
16	AM16	Address mask bit 16. Should be cleared.
17–31	—	Reserved, should be cleared

38.3 Address Map

Table 38-4 shows the base address map, while Table 38-5 provides the address map, including all registers in the execution units. The 17-bit SEC address bus value is shown. These address values are offsets from SECBR[BA] (refer to Section 38.2.1, “SEC Address Base Register”). Note that these tables show modulo-8 addresses; the 3 least-significant address bits that are used to select bytes within 64-bit-words are not shown.

Table 38-4. SEC Module Address Map

Address Offset[1] (AD 0–16)	Module	Description	Type	Reference
0x00000–0x00FFF	—	Reserved	—	—
0x01000–0x01FFF	Controller	Arbiter/controller control register space	Resource control	Section 38.7 on page 95
0x02000–0x02FFF	Channel_1	Crypto-channel 1	Data control	Section 38.6 on page 83
0x03000–0x03FFF	Channel_2	Crypto-channel 2		
0x04000–0x04FFF	Channel_3	Crypto-channel_3		
0x05000–0x05FFF	Channel_4	Crypto-channel_4		
0x06000–0x06FFF	—	Reserved	—	—
0x07000–0x07FFF	—	Reserved	—	—

Table 38-4. SEC Module Address Map (continued)

Address Offset[1] (AD 0–16)	Module	Description	Type	Reference
0x08000–0x08FFF	AFEU	Arc four execution unit	Crypto EU	Section 38.5.3 on page 45
0x0A000–0x0AFFF	DEU	Data encryption standard execution unit		Section 38.5.2 on page 36
0x0C000–0x0CFFF	MDEU	Message digest execution unit	Crypto EU	Section 38.5.4 on page 54
0x0E000–0x0EFFF	RNG	Random number generator		Section 38.5.5 on page 64
0x10000–0x10FFF	PKEU	Public key execution unit		Section 38.5.1 on page 28
0x12000–0x12FFF	AESU	Advanced encryption standard execution unit		Section 38.5.6 on page 70

[1] SEC register address offset is from SECBR. Refer to Section 38.2.1, “SEC Address Base Register."

Table 38-5 shows the system address map showing all functional registers.

Table 38-5. SEC Address Map

Address Offset[1] (AD 0–16)	Module	Register	Access	Reference
0x00800	—	Reserved	—	—
0x01000	Controller	EU assignment control	R/W	Section 38.7.1.1 on page 95
0x01008		Interrupt mask	R/W	Section 38.7.1.3 on page 96
0x01010		Interrupt status	R	Section 38.7.1.4 on page 97
0x01018		Interrupt clear	W	Section 38.7.1.5 on page 98
0x01020		Identification	R	Section 38.7.1.6 on page 100
0x01028		EU assignment status	R	Section 38.7.1.2 on page 96
0x01030		Master control	R	Section 38.7.1.7 on page 101
0x01038		Master error address	R	Section 38.7.1.8 on page 102
0x02008	Channel_1	Configuration register	R/W	Section 38.6.1.1 on page 84
0x02010		Pointer status	R	Section 38.6.1.2 on page 86
0x02040		Current descriptor pointer	R	Section 38.6.1.3 on page 91
0x02048		Fetch register	R/W	Section 38.6.1.4 on page 91
0x02080–0x020BF		Descriptor buffer[16]	R/W	Section 38.6.1.5 on page 92
0x03008	Channel_2	Config register	R/W	Section 38.6.1.1 on page 84
0x03010		Pointer status	R	Section 38.6.1.2 on page 86
0x03040		Current descriptor pointer	R	Section 38.6.1.3 on page 91
0x03048		Fetch register	R/W	Section 38.6.1.4 on page 91
0x03080–0x030BF		Descriptor buffer[16]	R/W	Section 38.6.1.5 on page 92

Table 38-5. SEC Address Map (continued)

Address Offset[1] (AD 0–16)	Module	Register	Access	Reference
0x04008	Channel_3	Config register	R/W	Section 38.6.1.1 on page 84
0x04010		Pointer status	R	Section 38.6.1.2 on page 86
0x04040		Current descriptor pointer	R	Section 38.6.1.3 on page 91
0x04048		Fetch register	R/W	Section 38.6.1.4 on page 91
0x04080–0x040BF		Descriptor buffer[16]	R/W	Section 38.6.1.5 on page 92
0x05008	Channel_4	Config register	R/W	Section 38.6.1.1 on page 84
0x05010		Pointer status	R	Section 38.6.1.2 on page 86
0x05040		Current descriptor pointer	R	Section 38.6.1.3 on page 91
050480x		Fetch register	R/W	Section 38.6.1.4 on page 91
0x05080–0x050BF		Descriptor buffer[16]	R/W	Section 38.6.1.5 on page 92
0x06000	—	Reserved	R/W	—
0x08000	AFEU	Mode register	R/W	Section 38.5.3.1 on page 45
0x08008		Key size register	R/W	Section 38.5.3.3 on page 46
0x08010		Data size register	R/W	Section 38.5.3.4 on page 47
0x08018		Reset control register	R/W	Section 38.5.3.5 on page 48
0x08028		Status register	R	Section 38.5.3.6 on page 49
0x08030		Interrupt status register	R	Section 38.5.3.7 on page 50
0x08038		Interrupt control register	R/W	Section 38.5.3.8 on page 51
0x08050		End of message register	W	Section 38.5.3.9 on page 53
0x08100–0x081FF		Context memory	R/W	Section 38.5.3.10.1 on page 53
0x08200		Context memory pointers	R/W	Section 38.5.3.10.2 on page 54
0x08400		Key register 0	W	Section 38.5.3.11 on page 54
0x08408		Key register 1	W	Section 38.5.3.11 on page 54
0x08800–0x08FFF		FIFO	R/W	Section 38.5.3.11.1 on page 54

Table 38-5. SEC Address Map (continued)

Address Offset[1] (AD 0–16)	Module	Register	Access	Reference
0x0A000	DEU	Mode register	R/W	Section 38.5.2.1 on page 36
0x0A008		Key size register	R/W	Section 38.5.2.2 on page 37
0x0A010		Data size register	R/W	Section 38.5.2.3 on page 38
0x0A018		Reset control register	R/W	Section 38.5.2.4 on page 38
0x0A028		Status register	R	Section 38.5.2.5 on page 39
0x0A030		Interrupt status register	R	Section 38.5.2.6 on page 40
0x0A038		Interrupt control register	R/W	Section 38.5.2.7 on page 42
0x0A050		EU-Go	W	Section 38.5.2.8 on page 44
0x0A100		IV register	R/W	Section 38.5.2.9 on page 44
0x0A400		Key 1 register	W	Section 38.5.2.10 on page 44
0x0A408		Key 2 register	W	Section 38.5.2.10 on page 44
0x0A410		Key 3 register	W	Section 38.5.2.10 on page 44
0x0A800–0x0AFFF		FIFO	R/W	Section 38.5.2.11 on page 44
0x0C000	MDEU	Mode register	R/W	Section 38.5.4.1 on page 54
0x0C008		Key size register	R/W	Section 38.5.4.3 on page 56
0x0C010		Data size register	R/W	Section 38.5.4.4 on page 57
0x0C018		Reset control register	R/W	Section 38.5.4.5 on page 57
0x0C028		Status register	R	Section 38.5.4.6 on page 58
0x0C030		Interrupt status register	R	Section 38.5.4.7 on page 59
0x0C038		Interrupt control register	R/W	Section 38.5.4.8 on page 60
0x0C050		EU_GO	W	Section 38.5.4.9 on page 62
0x0C100–0x0C120		Context memory	R/W	Section 38.5.4.10 on page 62
0x0C400–0x0C47F		Key memory	W	Section 38.5.4.11 on page 63
0x0C800–0x0CFFF		FIFO	W	Section 38.5.4.12 on page 63
0x0E000	RNG	Mode register	R/W	Section 38.5.5.1 on page 64
0x0E010		Data size register	R/W	Section 38.5.5.2 on page 65
0x0E018		Reset control register	R/W	Section 38.5.5.3 on page 65
0x0E028		Status register	R	Section 38.5.5.4 on page 66
0x0E030		Interrupt status register	R	Section 38.5.5.5 on page 67
0x0E038		Interrupt control register	R/W	Section 38.5.5.6 on page 68
0x0E050		EU_GO	W	Section 38.5.5.7 on page 69
0x0E800–0x0EFFF		FIFO	R	Section 38.5.5.8 on page 69

Table 38-5. SEC Address Map (continued)

Address Offset[1] (AD 0–16)	Module	Register	Access	Reference
0x10000	PKEU	Mode register	R/W	Section 38.5.1.1 on page 28
0x10008		Key size register	R/W	Section 38.5.1.2 on page 30
0x10010		Data size register	R/W	Section 38.5.1.3 on page 30
0x10018		Reset control register	R/W	Section 38.5.1.4 on page 30
0x10028		Status register	R	Section 38.5.1.5 on page 31
0x10030		Interrupt status register	R	Section 38.5.1.6 on page 32
0x10038		Interrupt control register	R/W	Section 38.5.1.7 on page 34
0x10050		EU_GO	W	Section 38.5.1.8 on page 35
0x10200–0x1023F		Parameter memory A0	R/W	Section 38.5.1.9 on page 35
0x10240–0x1027F		Parameter memory A1	R/W	
0x10280–0x102BF		Parameter memory A2	R/W	
0x102C0–0x102FF		Parameter memory A3	R/W	
0x10300–0x1033F		Parameter memory B0	R/W	
0x10340–0x1037F		Parameter memory B1	R/W	
0x10380–0x103BF		Parameter memory B2	R/W	
0x103C0–0x103FF		Parameter memory B3	R/W	
0x10400–0x104FF		Parameter memory E	W	
0x10800–0x108FF		Parameter memory N	R/W	
0x12000	AESU	Mode register	R/W	Section 38.5.6.1 on page 70
0x12008		Key size register	R/W	Section 38.5.6.2 on page 72
0x12010		Data size register	R/W	Section 38.5.6.3 on page 72
0x12018		Reset control register	R/W	Section 38.5.6.4 on page 73
0x12028		Status register	R	Section 38.5.6.5 on page 74
0x12030		Interrupt status register	R	Section 38.5.6.6 on page 75
0x12038		Interrupt control register	R/W	Section 38.5.6.7 on page 76
0x12050		End of message register	W	Section 38.5.6.8 on page 78
0x12100		Context	R/W	Section 38.5.6.9 on page 78
0x12400–0x12408		Key memory	R/W	Section 38.5.6.9.4 on page 82
0x12800–0x12FFF		FIFO	R/W	Section 38.5.6.9.5 on page 82

[1] SEC register address offset is from SECBR. Refer to Section 38.2.1, “SEC Address Base Register."

38.4 Data Packet Descriptors

The SEC has bus mastering capability to off load data movement and encryption operations from the host processor. As the system controller, 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 can either directly write keys, context, and data to the SEC (SEC in slave mode), or create a 'data packet descriptor' to guide the SEC through the security operation, with the SEC acting as a bus master. The descriptor can be created in main memory, any memory local to the SEC, or written directly to the data packet descriptor buffer in the SEC crypto-channel.

38.4.1 Descriptor Structure

SEC data packet descriptors are conceptually similar to descriptors used by most devices with DMA capability. See Figure 38-5 for a conceptual data packet descriptor. The descriptors are fixed length (64 bytes), and consist of 16 32-bit fields. Descriptors begin with a header that describes the security operation to be performed and the mode the execution unit will be set to while performing the operation.

The header is followed by seven data length/data pointer pairs. Data length indicates the amount of contiguous data to be transferred. This amount cannot exceed 32 Kbytes. The data pointer refers to the address of the data that the SEC fetches. Data in this case is broadly interpreted to mean keys, context, additional pointers, or the actual plain text to be permuted.

Figure 38-5 shows the data packet descriptor format. The descriptor consists of 16 32-bit fields. The number of fields provided in the descriptor allows for multi-algorithm operations require the fetch (and potentially return) of multiple keys and contexts. Any field that is not used is NULL, meaning it is filled with all zeros.

Bits	0	31	32 — 47	48 — 63	
Name	Header	DN	Reserved	Data Field 1 Length	Header/Len1
Reset	0	0	0	0	offset $080
Name	Data Field 1 Pointer		Reserved	Data Field 2 Length	Ptr1/Len2
Reset	0		0	0	offset $088
Name	Data Field 2 Pointer		Reserved	Data Field 3 Length	Ptr2/Len3
Reset	0		0	0	offset $090
Name	Data Field 3 Pointer		Reserved	Data Field 4 Length	Ptr3/Len4
Reset	0		0	0	offset $098
Name	Data Field 4 Pointer		Reserved	Data Field 5 Length	Ptr4/Len5
Reset	0		0	0	offset $0A0
Name	Data Field 5 Pointer		Reserved	Data Field 6 Length	Ptr5/Len6
Reset	0		0	0	offset $0A8
Name	Data Field 6 Pointer		Reserved	Data Field 7 Length	Ptr6/Len7
Reset	0		0	0	offset $0B0
Name	Data Field 7 Pointer		Next Data Packet Descriptor Pointer		Ptr7/NxtPtr
Reset	0		0		offset $0B8

Figure 38-5. Example Data Packet Descriptor

38.4.1.1 Descriptor Header

Descriptors are created by the host to guide the SEC through required crypto-graphic operations. The descriptor header defines the operations to be performed, 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 crypto-graphic operation. Figure 38-6 shows the descriptor header.

	0 11	12 23	24 27	28 29	30	31
Field	OP_0	OP_1	DESC_TYPE	—	ST	DN
Reset	0x0000_0000					
R/W	R/W					
Addr	Channel_1 0x02080, Channel_2 0x03080, Channel_3 0x04080, Channel_4 0x05080					

Figure 38-6. Descriptor Header

Table 38-6 defines the header bits.

Table 38-6. Header Bit Definitions

Bits	Name	Description
0–11	OP_0	OP_0 contains two subfields, EU_Select and Mode_Data. Figure 38-7 shows the subfield detail. EU_SELECT[0–3]—Programs the channel to select a primary EU of a given type. Table 38-7 lists the possible EU_SELECT values. MODE_DATA[4–11]— Programs the primary EU mode data. The mode data is specific to the chosen EU. This data is passed directly to bits 0–7 of the specified EU mode register.
12–23	OP_1	OP_1 contains two subfields, EU_Select and Mode_Data. Figure 38-7 shows the subfield detail. EU_SELECT[12–15]—Programs the channel to select a secondary EU of a given type. Table 38-7 lists the possible EU_SELECT values. MODE_DATA[16–23]—Programs the secondary EU mode data. The mode data is specific to the chosen EU. This data is passed directly to bits 0–7 of the specified EU mode register. **Note:** The MDEU is the only valid secondary EU. Values for Op1 EU_SELECT other than ‘MDEU’ or ‘No secondary EU selected’ will result in an ‘Unrecognized Header’ error condition. Selecting MDEU for both primary and secondary EU will also create an error condition.
24–27	DESC_TYPE	Descriptor type—Each type of descriptor determines the following attributes for the corresponding data length/pointer pairs: the direction of the data flow; which EU is associated with the data; and which internal EU address is used. Table 38-8 lists the valid types of descriptors.
28–29	—	Reserved. Should be cleared to zero
30	ST	Snoop type—Selects which of the two types of available snoop modes applies to the descriptor. 0 Snoop output data mode 1 Snoop input data mode In snoop input data mode, while the bus transaction to write data into the input FIFO of the primary EU is in progress, the secondary EU (always MDEU) will snoop the same data into its input FIFO. In snoop output data mode, the secondary EU (always MDEU) will snoop data into its input FIFO during the bus transaction to read data out of the output FIFO of the primary EU. **Note:** When snooping is not performed, this bit is ignored by the SEC crypto-channel.

Table 38-6. Header Bit Definitions (continued)

Bits	Name	Description
31	DN	DONE_NOTIFICATION_FLAG—Done notification flag. Setting this bit indicates whether to perform notification upon completion of this descriptor. The notification can take the form of an interrupt or modified header write back or both depending upon the state of the INTERRUPT_ENABLE and WRITEBACK_ENABLE control bits in the control register. 0 Do not signal DONE upon completion of this descriptor (unless globally programmed to do so through the master control register.) 1 Signal DONE upon completion of this descriptor **Note:** The SEC can be programmed to perform DONE notification upon completion of each descriptor, upon completion of any descriptor, or completion of a chain of descriptors. This bit provides for the second case.

Figure 38-7 shows the two sub fields of OP_*n*.

0 3	4 11
OP_*n*	
EU_SELECT	MODE_DATA

Figure 38-7. OP_*n* sub fields

OP0 EU_SELECT values of ‘no EU selected’ or ‘reserved EU’ result in an ‘unrecognized header error’ condition during processing of the descriptor header. Also, the primary EU selected by the OP0 EU_SELECT field may only be DEU, AESU or AFEU when a valid secondary EU is selected. For this case, all other values of OP0 EU_SELECT result in an ‘unrecognized header’ error condition. The full range of permissible EU_Select values is shown in Table 38-7.

Table 38-7. EU_Select Values

Value	EU Select
0000	No EU selected.
0001	AFEU
0010	DEU
0011	MDEU
0100	RNG
0101	PKEU
0110	AESU
Others	Reserved

Table 38-8 shows the permissible values for the descriptor type field in the descriptor header.

Table 38-8. Descriptor Types

Value	Descriptor Type	Notes
0000	aesu_ctr_nonsnoop	AESU CTR nonsnooping
0001	common_nonsnoop_no_afeu	Common, nonsnooping, non-PKEU, non-AFEU
0010	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011	non_hmac_snoop_no_afeu	Snooping, non-HMAC, non-AFEU
0100	aseu_key expand_output	Non-snooping, non HMAC, AESU, expanded key out
0101	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110	hmac_snoop_afeu	Snooping, HMAC, AFEU (no context out)
0111	non_hmac_snoop_afeu	Snooping, non-HMAC, AFEU
1000	pkeu_mm	PKEU-MM
1001	pkeu_ec	PKEU-EC
1010	pkeu_static_ec_point	PKEU static-EC point (completes operand loading and executes)
1011	pkeu_static_ec_parameter	PKEU Static-EC Parameter (preloads EC operands)
1100	hmac_snoop_aesu_ctr	AESU CTR hmac snooping
1101	non_hmac_snoop_aesu_ctr	AESU CTR non-hmac snooping
1110	hmac_snoop_afeu_ key_in	AFEU context out available
1111	hmac_snoop_afeu_ctx_in	AFEU Context out available

38.4.1.2 Descriptor Length and Pointer Fields

The length and pointer fields represent one of seven data length/pointer pairs. Each pair defines a block of data in system memory. The length field gives the length of the block in bytes. The maximum allowable number of bytes is 32 Kbytes. A value of zero loaded into the length field indicates that this length/pointer pair should be skipped and processing continue with the next pair.

The pointer field contains the address, in global memory, of the first byte of the data block. Transfers with the pointer address set to zero will have the length value written to the EU, and no data fetched from the memory.

NOTE

Certain public key operations require information about data length, but not the data itself. Figure 38-8 shows the descriptor length field.

	32	47	48	63
Field	—		DATA LENGTH	
	0		msb<--------lsb	
Reset	0			
R/W	R/W			

Figure 38-8. Descriptor Length Field

Table 38-9 shows the descriptor length field mapping.

Table 38-9. Descriptor Length Field Mapping

Bits	Name	Reset Value	Description
32–47	—	0	Reserved, set to zero
48–63	DATA LENGTH	0	**Note:** The maximum length this field can be set to 32 Kbytes. Under host control, a channel can be temporarily locked static, and data only” descriptors can be chained to fetch blocks larger than 32 Kbytes in 32-Kbyte sub-blocks without key/context switching, until the large original block has been completely ciphered. Length fields also indicate the size of items to be written back to memory upon completion of security processing in the SEC.

Figure 38-9 shows the descriptor pointer field.

	0 … 31
Field	DATA FIELD X POINTER
Reset	0
R/W	R/W

Figure 38-9. Descriptor Pointer Field

Table 38-10 shows the descriptor pointer field mapping.

Table 38-10. Descriptor Pointer Field Mapping

Bits	Name	Reset Value	Description
0–31	DATA FIELD POINTER	0	The data pointer field contains the address, in global memory, of the first byte of the data packet for either read or write back. Transfers from the bus with pointer address set to zero will be skipped.

Table 38-11 shows how the length/pointer pairs should be used with the various descriptor types to load keys, context, and data into the execution units, and how the required outputs should be unloaded.

Table 38-11. Descriptor Length/Pointer Mapping

Descriptor	L/P 1	L/P 2	L/P 3	L/P 4	L/P 5	L/P 6	L/P 7
0000	Null	IV	Key	Data in	Data out	IV out	MAC out
0001	Nil	IV	Key	Data in	Data out	IV out	MAC out
0010	HMAC Key	HMAC Data	Key	IV	Data in	Data out	HMAC/Context out
0011	MD Ctx In	IV	Key	Data in	Data out	IV out	MD/Context out
0100	Nil	IV	Key	Data in	Data out	IV out	Key out through FIFO
0101	Nil	In via FIFO context	Key	Data in	Data out	Out via FIFO context	Nil
0110	HMAC Key	HMAC Data	Key	IV in through FIFO	Data in	Data out	HMAC/Context out

Table 38-11. Descriptor Length/Pointer Mapping (continued)

Descriptor	L/P 1	L/P 2	L/P 3	L/P 4	L/P 5	L/P 6	L/P 7
0111	MD Ctx In	IV in through FIFO	Key	Data in	Data out	IV out through FIFO	MD/Context out
1000	B	A	E	N	B out	Nil	Nil
1001	B	A	Key	N	B1 out	Nil	Nil
1010	A0	A1	A2	B1 out	B2 out	B3 out	Nil
1011	A3	B0	B1	Key	N	Nil	Nil
1100	HMAC key	HMAC data	Key	IV	Data in	Data out	HMAC/Context out
1101	MD Ctx in	IV	Key	Data in	Data out	IV out	MD/Context out
1110	HMAC key	HMAC data	Key	Data in	Data out	IV out through FIFO	HMAC/Context out
1111	HMAC key	HMAC data	IV	Data in	Data out	IV out through FIFO	HMAC/Context out

38.4.2 Descriptor Chaining

Following the length/pointer pairs is the ‘Next Descriptor’ field, which contains the pointer to the next descriptor in memory. Upon completion of processing of the current descriptor, this value, if non-zero, is used to request a burst read of the next-data-packet descriptor. This automatic load of the next descriptor is referred to as descriptor chaining. Figure 38-10 displays the next descriptor pointer field.

	0 — 31
Field	NEXT DESCRIPTOR POINTER
Reset	0
R/W	R/W

Figure 38-10. Next Descriptor Pointer Field

Table 38-12 describes the descriptor pointer field mapping.

Table 38-12. Descriptor Pointer Field Mapping

Bits	Name	Reset Value	Description
0–31	NEXT DESCRIPTOR POINTER	0	The next descriptor pointer field contains the address, in global memory space, of the next descriptor to be fetched if descriptor chaining is enabled.

Descriptor chaining provides a measure of ‘decoupling’ between host CPU activities and the status of the SEC. Rather than waiting for the SEC to signal DONE, and arbitrating for the bus in order to write directly to the next-data-packet descriptor in the crypto-channel, the host can simply create new descriptors in memory, and chain them to descriptors which have not yet been fetched by the SEC by filling the next-data-packet field with the address of the newly created descriptor. Whether or not processing continues automatically following next-descriptor fetch and whether or not an interrupt is generated depends on the programming of the crypto-channel's configuration register.

See Section 38.6.1.1, "Crypto-Channel Configuration Register (CCCR)," for additional information on how the SEC can be programmed to signal and act upon completion of a descriptor.

NOTE

> It is possible to insert a descriptor into an existing chain; however, great care must be taken when doing so.

Figure 38-11 shows a conceptual chain, or 'linked list,' of descriptors.

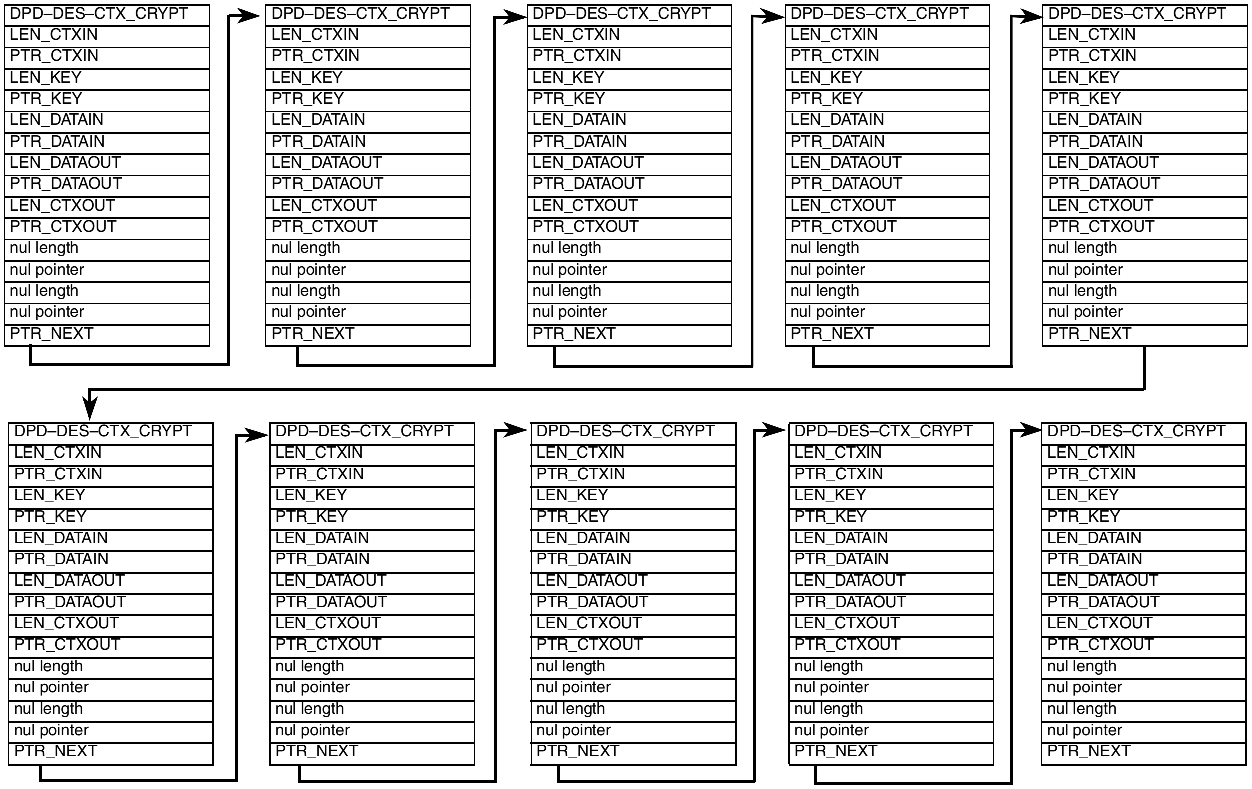


Figure 38-11. Chain of Descriptors

38.4.2.1 Null Fields

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 will be required to load the required keys, context, and data. (Some operations do not require context, others may only need to fetch a small, contiguous block of data.) Therefore, when processing data packet descriptors, the SEC will skip entirely any pointer that has an associated length of zero.

38.4.3 Descriptor Classes

The SEC has two general classes of descriptors: static, which refers to a relatively unchanging usage of the SEC resources, and dynamic, which refers to a continually changing usage model.

38.4.3.1 Static Descriptors

Recall that the SEC has six execution units and four crypto-channels. The EUs can be statically assigned or dedicated to a particular crypto-channel. Certain combinations of EUs can be statically assigned to the same crypto-channel to facilitate multi-operation security processes, such as IPSec ESP mode. When the system traffic model permits its use, static assignment can offer significant performance improvements over dynamic assignment by avoiding key and context switching per packet.

Static descriptors split the operations to be performed during a security operation into separate descriptors. The first descriptor is typically used to set the EU mode, load the key and context, and to read/permute/write the first block of data. The second (and multiple subsequent) descriptor contains length/pointer pairs to the remaining data to be permuted. The final descriptor read/permute/writes the final block of data, and outputs any context that needs to be preserved for later use. Because the key and context are unchanging over multiple packets (or descriptors), the series of short reads and writes required to set up and tear down a session are avoided. This savings, along with the crypto-channel having dedicated execution units, represents a noticeable performance improvement.

NOTE

> In most cases, static descriptors are the same as dynamic descriptors, but with lengths and pointers for context out omitted. Static assignment is achieved through the controller’s EUACR (Figure 38-67).

For example, statically assigning AFEU to a particular crypto-channel permits AFEU to retain state between data packets. The following descriptors, displayed in Table 38-13 through Table 38-16, support state-retention. Table 38-13 defines the common_nonsnoop_afeu descriptor.

Table 38-13. Descriptor common_nonsnoop_afeu

Field	Value/ Type	Description
common_nonsnoop_afeu [1]	0x100000050	AFEU, new key, do not dump context, perform permute
LEN_1	Length	Place holder
PTR_1	Pointer	Place holder
LEN_2	Length	Place holder
PTR_2	Pointer	Place holder
LEN_3	Length	Length of ARC-4 key
PTR_3	Pointer	Pointer to ARC-4 key
LEN_4	Length	Length of data to be read and permuted
PTR_4	Pointer	Pointer to data in memory
LEN_5	Length	Length of data to be written after permutation
PTR_5	Pointer	Pointer to memory buffer for write back

Table 38-13. Descriptor common_nonsnoop_afeu (continued)

Field	Value/ Type	Description
LEN_6	Length	Unused
PTR_6	Pointer	Unused
LEN_7	Length	Unused
PTR_7	Pointer	Unused
PTR_NEXT	Pointer	Pointer to next descriptor

[1] The common_nonsnoop_afeu descriptor writes the key at PTR_3 to be written to AFEU and causes the initial context-permutation to occur.

Table 38-14 shows the common_nonsnoop_afeu descriptor continuing data permutation with the ARC-4 context (s-box) created in the previous descriptor.

Table 38-14. Continuation of common_nonsnoop_afeu

Field	Value / Type	Description
common_nonsnoop_afeu	0x101000050	AFEU, reuse context, do not dump context, prevent permute
LEN_1	Length	Place holder
PTR_1	Pointer	Place holder
LEN_2	Length	Place holder
PTR_2	Pointer	Place holder
LEN_3	Length	Place holder
PTR_3	Pointer	Place holder
LEN_4	Length	Length of data to be read and permuted
PTR_4	Pointer	Pointer to data in memory
LEN_5	Length	Length of data to be written after permutation
PTR_5	Pointer	Pointer to memory buffer for write back
LEN_6	Length	Unused
PTR_6	Pointer	Unused
LEN_7	Length	Unused
PTR_7	Pointer	Unused
PTR_NEXT	Pointer	Pointer to next descriptor

Table 38-15 shows the common_nonsnoop_afeu descriptor continuing data permutation with the ARC-4 context (s-box), and outputting the context following completion of the last block of data.

Table 38-15. Wrap-up of common_nonsnoop_afeu

Field	Value / Type	Description
common_nonsnoop_afeu	0x103000050	AFEU, reuse context, dump context, prevent permute
LEN_1	Length	Place holder
PTR_1	Pointer	Place holder
LEN_2	Length	Place holder
PTR_2	Pointer	Place holder
LEN_3	Length	Place holder
PTR_3	Pointer	Place holder
LEN_4	Length	Length of data to be read and permuted
PTR_4	Pointer	Pointer to data in memory
LEN_5	Length	Length of data to be written after permutation
PTR_5	Pointer	Pointer to memory buffer for write back
LEN_6	Length	Length of S-Box context to be written after permutation (always 259 bytes)
PTR_6	Pointer	Pointer to memory for write back of S-box context
LEN_7	Length	Unused
PTR_7	Pointer	Unused
PTR_NEXT	Pointer	Pointer to next descriptor

Table 38-16 shows how the common_nonsnoop_afeu descriptor can resume permutation with an existing context, rather than generating a context directly from a key. This descriptor can be considered an alternate to the descriptor shown in Table 38-13. Table 38-14 and Table 38-15 can follow Table 38-16, just as they did Table 38-13.

Table 38-16. Descriptor common_nonsnoop_afeu

Field	Value / Type	Description
common_nonsnoop_afeu[1]	0x105000050	AFEU, context from FIFO, do not dump context, prevent permute
LEN_1	Length	Place holder
PTR_1	Pointer	Place holder
LEN_2	Length	Length of ARC-4 context (always 259 bytes)
PTR_2	Pointer	Pointer to ARC-4 context
LEN_3	Length	Place holder
PTR_3	Pointer	Place holder
LEN_4	Length	Length of data to be read and permuted
PTR_4	Pointer	Pointer to data in memory

Table 38-16. Descriptor common_nonsnoop_afeu (continued)

Field	Value / Type	Description
LEN_5	Length	Length of data to be written after permutation
PTR_5	Pointer	Pointer to memory buffer for write back
LEN_6	Length	Unused
PTR_6	Pointer	Unused
LEN_7	Length	Unused
PTR_7	Pointer	Unused
PTR_NEXT	Pointer	Pointer to next descriptor

[1] common_nonsnoop_afeu fetches an existing context through the AFEU Input FIFO and uses this context to permute the first block of data.

38.4.3.2 Dynamic Descriptors

In a typical networking environment, packets from innumerable sessions can arrive randomly. The host must determine which security association applies to the current packet and encrypt or decrypt without any knowledge of the security association of the previous or next packet. This situation calls for the use of dynamic descriptors.

When under dynamic assignment, an EU must be used under the assumption that a different crypto-channel (with a different context) may have just used the EU and that another crypto-channel (with yet another context) may use that EU immediately after the current crypto-channel has released the EU. Therefore, for dynamic-assignment use, there is a set of data packet descriptors defined that sets up the appropriate context, performs the cipher function, and saves the context to system memory.

The descriptor shown in Table 38-17 completely sets up the DEU and MDEU for an encryption operation, loads the HMAC and symmetric keys, context, and data, writes the permuted data back to memory, and writes the HMAC and any altered context (IV) back to memory. (This may be necessary when DES is operating in CBC mode with implicit IV.) Upon completion of the descriptor, the DEU and MDEU are cleared and released.

Table 38-17. Descriptor HMAC_Snoop_Non_AFEU

Field	Value / Type	Description
HMAC_Snoop_Non_AFEU	0x2073FC20	Typical IPSec descriptor. With 3DES-HMAC-SHA-1.
LEN_1	Length	Number of bytes of HMAC key to be written
PTR_1	Pointer	Pointer to HMAC key
LEN_2	Length	Number of bytes data to be processed for HMAC
PTR_2	Pointer	Pointer to start of HMAC data
LEN_3	Length	Number of bytes of symmetric key
PTR_3	Pointer	Pointer to symmetric key
LEN_4	Length	Number of bytes of symmetric IV

Table 38-17. Descriptor HMAC_Snoop_Non_AFEU (continued)

Field	Value / Type	Description
PTR_4	Pointer	Pointer to symmetric IV
LEN_5	Length	Length of data to be read and permuted
PTR_5	Pointer	Pointer to data in memory
LEN_6	Length	Length of data to be written after permutation
PTR_6	Pointer	Pointer to memory buffer for write back
LEN_7	Length	Length of HMAC to be written to memory
PTR_7	Pointer	Pointer to memory buffer for HMAC
PTR_NEXT	Pointer	Pointer to next descriptor

38.5 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 Rinjdael symmetric key cipher
- ARC four execution unit (AFEU)
- Message digest execution unit (MDEU)
- A 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.

NOTE

The execution units are natively little endian. Register values are shown in a big-endian format to assist in debug in a 60x (big-endian) environment. Much of the following details are required only for debug and operation of the SEC in slave mode. When operating as an initiator, the device drivers abstract register-level operations, and the crypto-channels and controller operate the execution units.

38.5.1 Public Key Execution Units (PKEUs)

This section contains details about the public key execution units (PKEUs), including a detailed register map and the modes of operation, status and control registers, and parameter RAMs. The following sections describe PKEU registers and parameter memories.

NOTE

> The SEC 1.0 PKEU is designed to operate on big numbers, represented in strings up to 2048 bits long. The PKEU's internal architecture is natively 64-bit little-endian and expects data least significant word first. Input data (exponents, modulus, etc.) should be represented with the least significant bit aligned to the right, then be both byte and word swapped in memory for proper handling by the big endian core of the MPC8272. For example, a string of 0x12abcdef 123456789abcdef0f1000000 should be represented as an integer of 0x00000012abcdef123456789abcdef0f1 and appear in memory as 0xf1f0debc9a78563412 efcdab12000000.

38.5.1.1 PKEU Mode Register

This register specifies the internal PKEU routine to be executed. For the root arithmetic routines, PKEU has the capability to perform arithmetic operations on subsegments of the entire memory. This is particularly useful for operations such as ECDH (elliptic curve Diffie-Hellman) key agreement computation. By using regAsel and regBsel, for example, parameter memory A subsegment 2 can be multiplied into parameter memory B subsegment 1. Figure 38-12 and Figure 38-13 detail two definitions.

	0 7	8 63
Field	MODE	—
Reset	0	0
R/W	R/W	
Addr	PKEU 0x10000	

Figure 38-12. PKEU Mode Register: Definition 1

	0 3	4 7	8 63
Field	MODE	REGSEL	—
Reset	0	0	0
R/W	R/W		
Addr	PKEU 0x10000		

Figure 38-13. PKEU Mode Register: Definition 2

Table 38-18 lists the possible values. The value written to PKEU[Mode] determines the routine used. Parameter memories are referred to for the base address, as shown.

Table 38-18. PKEU Mode Register Field Description

Routine	Description	Mode[1]		
		[0–3]	[4–5]	[6–7]
—	Reserved	0000	00	00
CLEARMEMORY	Clear memory	0000	0	01
MOD_EXP	FP: Exponentiate mod N and deconvert from Montgomery format	0000	00	10
MOD_R2MODN	FP: Compute Montgomery converter (R2 mod N)	0000	00	11
MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem (RnRp mod N)	0000	01	00
EC_FP_AFF_PTMULT	FP EC: Multiply key times point in affine coordinates	0000	01	01
EC_F2M_AFF_PTMULT	F2m EC: Multiply key times point in affine coordinates	0000	01	10
EC_FP_PROJ_PTMULT	FP EC: Multiply key times point in projective coordinates[2]	0000	01	11
EC_F2M_PROJ_PTMULT	F2m EC: Multiply key times point in projective coordinates[2]	0000	10	00
EC_FP_ADD	FP EC: Add two points in projective coordinates[2]	0000	10	01
EC_FP_DOUBLE	FP EC: Double a point in projective coordinates[2]	0000	10	10
EC_F2M_ADD	F2m EC: Add two points in projective coordinates[2]	0000	10	11
EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates[2]	0000	11	00
F2M_R2	F2m: Compute Montgomery converter (R2 mod N)	0000	11	01
F2M_INV	F2m: Invert mod N	0000	11	10
MOD_INV	FP: Invert mod N	0000	11	11
MOD_ADD	FP: Add mod N	0001	regAsel[3] 00 = A0 01 = A1 10 = A2 11 = A3	regBsel[3] 00 = B0 01 = B1 10 = B2 11 = B3
MOD_SUB	FP: Subtract mod N	0010		
MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format	0011		
MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format	0100		
POLY_F2M_ADD	F2m: Add mod N	0101		
POLY_F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format	0110		
POLY_F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format	0111		
RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, POLY_F2M_MULT1_MONT, and MOD_EXP)	1000	00	00

[1] As shown in Figure 38-13, bits 4–7 are defined as REGSEL for definition 2.

[2] The input Z is assumed to be nonzero. If Z = 0, then the result of the point multiply in projective coordinates (undefined, undefined, 0) is returned.

[3] regAsel and regBsel here refer to the specific segment of parameter memory A and B.

38.5.1.2 PKEU Key Size Register

The key size register reflects the number of significant bytes to be used from PKEU Parameter Memory E in performing modular exponentiation or elliptic curve point multiplication. The minimum value for this register, when performing either modular exponentiation or elliptic curve point multiplication, is 1 byte. (0–15 = 0x0100). The maximum legal value is 256 bytes. (0–15 = 0x0001). To avoid a key size error, 12–14 must be set to zero, and the value of [0–7, 15] must not be greater than 256.

	0 7	8 14	15	16 63
Field	Key Size	—	Key Size	—
	<----lsb		msb	
Reset	0			
R/W	R/W			
Addr	PKEU 0x10008			

Figure 38-14. PKEU Key Size Register

38.5.1.3 PKEU Data Size Register

The PKEU data size register, shown in Figure 38-15, specifies, in bits, the size 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, 160 bits,...), will be represented internally as the same value (128 bits, 160 bits,...). Any value written that is not a multiple of 32 bits (for example, 132 bits, 161 bits,...), will be represented internally as the next larger 32-bit multiple (160 bits, 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). (0:15 = 0x6100) The maximum size to operate properly is 2048 bits (0:15 = 0x0008). A value in bits larger than 2048 will result in a data size error.

	0 7	8 11	12 15	16 63
Field	Data Size	—	Data Size	—
	<----lsb		msb<-----	
Reset	0			
R/W	R/W			
Addr	PKEU 0x10010			

Figure 38-15. PKEU Data Size Register

38.5.1.4 PKEU Reset Control Register

This register, Figure 38-16, contains three reset options specific to the PKEU.

	0 4	5	6	7	8 63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	PKEU 0x10018				

Figure 38-16. PKEU Reset Control Register

Table 38-19 describes the PKEU reset control register’s fields.

Table 38-19. PKEU Reset Control Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RI	Reset interrupt. Writing this bit active high causes PKEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the PKEU interrupt status register. 0 Do not reset. 1 Reset interrupt logic
6	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 RESET_DONE bit in the PKEU status register (Section 38.5.1.5, “PKEU Status Register”). 0 Do not reset. 1 Reset most of PKEU
7	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 RESET_DONE bit in the PKEU status register will indicate when this initialization routine is complete (Section 38.5.1.5, “PKEU Status Register”). 0 Do not reset. 1 Full PKEU reset
8–63	—	Reserved

38.5.1.5 PKEU Status Register

This status register contains 5 bits that reflect the state of PKEU internal fields.

The PKEU status register is read-only. Writing to this location will result in address error being reflected in the PKEU interrupt status register.

	0	1	2	3	4	5	6	7	8	63
Field	—	Z	HALT	—		IE	ID	RD	—	
Reset	0	0	0	0		0	0	0	0	
R/W	R									
Addr	PKEU 0x10028									

Figure 38-17. PKEU Status Register

Table 38-20 describes the PKEU status register's fields.

Table 38-20. PKEU Status Register Field Descriptions

Bits	Name	Description
0	—	Reserved
1	Z	Zero. This bit 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.
2	HALT	Halt indicates that the PKEU has halted due to an error. 0 PKEU not halted 1 PKEU halted **Note:** Because the error causing the PKEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
3–4	—	Reserved
5	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 PKEU is not signaling error 1 PKEU is signaling error
6	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 PKEU is not signaling done 1 PKEU is signaling done
7	RD	Reset done. This status bit, when high, indicates that PKEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8–63	—	Reserved **Note:** Some bits in the upper portion of this register are used as state tables for internal PKEU routines. In order to avoid confusion should the user read this register during normal operation, the user is advised that these bits exist, but their specific definition is reserved.

38.5.1.6 PKEU Interrupt Status Register

The interrupt status register tracks the state of possible errors—if those errors are not masked—through the PKEU interrupt control register. The definition of each bit in the PKEU interrupt status register is shown in Figure 38-18.

	0	1	2 9	10	11	12	13	14	15	16 63
Field	ME	AE	—	INV	IE	—	CE	KSE	DSE	—
Reset	0	0	0	0	0	0	0	0	0	0
R/W	R									
Addr	PKEU 0x10030									

Figure 38-18. PKEU Interrupt Status Register

Table 38-21 describes PKEU interrupt status register fields.

Table 38-21. PKEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0	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.
1	AE	Address error. Illegal read or write address was detected within the PKEU address space. 0 No error detected 1 Address error
2–9	—	Reserved
10	INV	Inversion error. Indicates that the inversion routine has a zero operand. 0 No inversion error detected 1 Inversion error detected
11	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.
12	—	Reserved
13	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
14	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
15	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
16–63	—	Reserved

38.5.1.7 PKEU Interrupt Control Register

The PKEU interrupt control register controls the result of detected errors. For a given error (as defined in Section 38.5.1.6, “PKEU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the PKEU interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2 9	10	11	12	13	14	15	16 63
Field	ME	AE	—	INV	IE	—	CE	KSE	DSE	—
Reset	0	0	0	0	0	0	0	0	0	0
R/W	R/W									
Addr	PKEU 0x10038									

Figure 38-19. PKEU Interrupt Control Register

Table 38-22 describes PKEU interrupt control register fields.

Table 38-22. PKEU Interrupt Control Register Field Descriptions

Bits	Name	Description
0	ME	Mode error 0 Mode error enabled 1 Mode error disabled
1	AE	Address error 0 Address error enabled 1 Address error disabled
2–9	—	Reserved
10	INV	Inversion error 0 Inversion error enabled 1 Inversion error disabled
11	IE	Internal error 0 Internal error enabled 1 Internal error disabled
12	—	Reserved
13	CE	Context error 0 Context error enabled 1 Context error disabled
14	KSE	Key size error 0 Key size error enabled 1 Key size error disabled
15	DSE	Data size error 0 Data size error enabled 1 Data size error disabled
16–63	—	Reserved

38.5.1.8 PKEU EU_GO Register

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. The PKEU EU_GO register is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the PKEU (through an internally generated write to the EU_GO register) when the SEC acts as an initiator.

	0	63
Field	PKEU EU_GO	
Reset	0	
R/W	W	
Addr	PKEU 0x10050	

Figure 38-20. PKEU EU_GO Register

38.5.1.9 PKEU Parameter Memories

The PKEU uses four 2048-bit memories to receive and store operands for the arithmetic operations the PKEU is 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.

38.5.1.9.1 PKEU Parameter Memory A

This 2048-bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 512-bit memories and is used to specify particular curve parameters and input values.

38.5.1.9.2 PKEU Parameter Memory B

This 2048-bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 512-bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.

38.5.1.9.3 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 causes an address error to be reflected in the PKEU interrupt status register.

38.5.1.9.4 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.

38.5.2 Data Encryption Standard Execution Units (DEUs)

This section contains details about the data encryption standard execution units (DEUs), including a detailed register map and the 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 the on-chip controller will abstract register level access from the user.

38.5.2.1 DEU Mode Register

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 as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the DEU is reset or reinitialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error is generated.

	0 4	5	6	7	8 12	13 15	16 63
Field	—	CE	TS	ED	—	BURST SIZE	—
Reset	0	0	0	0	0	0	0
R/W	R/W						
Addr	DEU 0x0A000						

Figure 38-21. DEU Mode Register

Table 38-23 describes DEU mode register fields.

Table 38-23. DEU Mode Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	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

Table 38-23. DEU Mode Register Field Descriptions (continued)

Bits	Name	Description
6	TS	Triple/Single DES. If set, DEU operates the Triple DES algorithm; if not set, DEU operates the single DES algorithm. 0 Single DES 1 Triple DES
7	ED	Encrypt/Decrypt. If set, DEU operates the encryption algorithm; if not set, DEU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption
8–12	—	Reserved
13–15	BURST SIZE	The 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 crypto-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 DEU 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 DEU.
16–63	—	Reserved

38.5.2.2 DEU Key Size Register

This value indicates the number of bytes of key memory that should be used in encrypting or decrypting. If the DEU mode register is set for single DES, any value other than 8 bytes will automatically generate a key size error in the DEU interrupt status register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1 = K3) or 24 bytes (168 bits for three-key triple DES) will generate an error. Triple DES always uses K1 to encrypt, Key2 to decrypt, K3 to encrypt.

	0 1	2 7	8 63
Field	—	Key Size	—
		msb<----lsb	
Reset	0	0	0
R/W	R/W		
Addr	DEU 0x0A008		

Figure 38-22. DEU Key Size Register

Table 38-24 shows the legal values for DEU key size.

Table 38-24. DEU Key Size Register Field Descriptions

Bits	Name	Description
0–7	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)
7–63	—	Reserved

38.5.2.3 DEU Data Size Register

This register, shown in Figure 38-23, 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 the DEU data size register, however only bits 2–7 are checked to determine if there is a data size error. If 2–7 are all zeros, the message is evenly divisible into 64-bit blocks. In slave mode, the user must write the data size to the data size register. If the data size written is not divisible by 64-bits (2–7 non-zero), a data size error will occur.

	0 1	2 7	8 63
Field	—	Data Size	—
		msb<----lsb	
Reset	0		
R/W	R/W		
Addr	DEU 0x0A010		

Figure 38-23. DEU Data Size Register

38.5.2.4 DEU Reset Control Register

This register, shown in Figure 38-24, allows three levels reset of just DEU, as defined by the three self-clearing bits:

	0 4	5	6	7	8 63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	DEU 0x0A018				

Figure 38-24. DEU Reset Control Register

Table 38-25 describes DEU reset control register fields.

Table 38-25. DEU Reset Control Register Field Descriptions

Bits	Names	Description
0–4	—	Reserved
5	RI	Reset interrupt. Writing this bit active high causes DEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the DEU interrupt status register. 0 Do not reset 1 Reset interrupt logic

Table 38-25. DEU Reset Control Register Field Descriptions (continued)

Bits	Names	Description
6	MI	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the DEU status register 0 Do not reset 1 Reset most of DEU
7	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the DEU status register will indicate when this initialization routine is complete 0 Do not reset 1 Full DEU reset
8–63	—	Reserved

38.5.2.5 DEU Status Register

This status register, displayed in Figure 38-25, contains 6 bits that reflect the state of DEU internal signals. The DEU status register is read-only. Writing to this location will result in address error being reflected in the DEU interrupt status register.

	0 1	2	3	4	5	6	7	8 63
Field	—	HALT	IFW	OFR	IE	ID	RD	—
Reset	0	0	0	0	0	0	0	0
R/W	R							
Addr	DEU 0x0A028							

Figure 38-25. DEU Status Register

Table 38-20 describes the DEU status register’s bit settings.

Table 38-26. DEU Status Register

Bits	Name	Description
0–1	—	Reserved
2	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.

Table 38-26. DEU Status Register (continued)

Bits	Name	Description
3	IFW	Input FIFO writable. The controller uses this signal to determine if the DEU can accept the next burst size block of data. 0 DEU Input FIFO not ready 1 DEU Input FIFO ready **Note:** The 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 crypto-channel that a 'burst size' amount of space is available in the FIFO. The documentation of this bit in the DEU status register is to avoid confusing a user who may read this register in debug mode.
4	OFR	Output FIFO readable. The controller uses this signal to determine if the DEU can source the next burst size block of data. 0 DEU Output FIFO not ready 1 DEU Output FIFO ready **Note:** The 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 crypto-channel that a 'burst size' amount of data is available in the FIFO. The documentation of this bit in the DEU status register is to avoid confusing a user who may read this register in debug mode.
5	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 DEU is not signaling error 1 DEU is signaling error
6	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 DEU is not signaling done 1 DEU is signaling done
7	RD	Reset done. This status bit, when high, indicates that DEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8–63	—	Reserved

38.5.2.6 DEU Interrupt Status Register

The DEU interrupt status register, shown in Figure 38-26, tracks the state of possible errors, if those errors are not masked, through the DEU interrupt control register.

	0	1	2	3	4	5	6	7 9	10	11	12	13	14	15	16 63
Field	ME	AE	OFE	IFE	—	IFO	OFU	—	KPE	IE	ERE	CE	KSE	DSE	—
Reset	0														
R/W	R														
Addr	DEU 0x0A030														

Figure 38-26. DEU Interrupt Status Register

Table 38-27 describes DEU interrupt register fields.

Table 38-27. DEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0	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
1	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
2	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
3	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
4	—	Reserved
5	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 slave, the SEC cannot accept FIFO inputs larger than 512 bytes without overflowing.
6	OFU	Output FIFO underflow. The DEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7–9	—	Reserved
10	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
11	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error **Note:** This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the interrupt control register or by resetting the DEU.
12	ERE	Early read error. The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
13	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

Table 38-27. DEU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
14	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
15	DSE	Data size error (DSE): A value was written to the DEU data size register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
16–63	—	Reserved

38.5.2.7 DEU Interrupt Control Register

The interrupt control register controls the result of detected errors. For a given error (as defined in Section 38.5.2.6, “DEU Interrupt Status Register”), 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.

	0	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	63
Field	ME	AE	OFE	IFE	—	IFO	OFU	—		KPE	IE	ERE	CE	KSE	DSE	—	
Reset	0																
R/W	R/W																
Addr	DEU 0x0A038																

Figure 38-27. DEU Interrupt Control Register

Table 38-28. DEU Interrupt Control Register Field Descriptions

Bits	Name	Description
0	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
2	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

Table 38-28. DEU Interrupt Control Register Field Descriptions

Bits	Name	Description
3	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
4	—	Reserved
5	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 slave, the SEC cannot accept FIFO inputs larger than 512 bytes without overflowing.
6	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
7–9	—	Reserved
10	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
11	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
12	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
13	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
14	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
15	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
16–63	—	Reserved

38.5.2.8 DEU EU_GO Register

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, 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 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 this register is merely a trigger causing the DEU 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 DEU (through an internally generated write to the EU_Go register) when the SEC acts as an initiator.

	0	63
Field	DEU EU_GO	
Reset	0	
R/W	W	
Addr	DEU 0x0A050	

Figure 38-28. DEU EU_GO Register

38.5.2.9 DEU IV Register

For CBC mode, the initialization vector is written to and read from the DEU IV register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

38.5.2.10 DEU Key Registers

The DEU uses three write-only key registers to perform encryption and decryption. In single DES mode, only key register 1 may be written. The value written to key register 1 is simultaneously written to key register 3, auto-enabling the DEU for 112-bit triple DES if the key size register indicates 2-key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit triple DES, key register 1 must be written first, followed by the write of key register 2, the key register 3.

Reading any of these memory locations generates an address error interrupt.

38.5.2.11 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 of the DEU output FIFO. Overflows and underflows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

38.5.3 ARC Four Execution Units (AFEUs)

This section contains details about the ARC four execution units (AFEUs), including a detailed register map and the 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.

38.5.3.1 AFEU Mode Register

Shown in Figure 38-29, the AFEU mode register contains 3 bits that 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 as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the AFEU 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.

38.5.3.2 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 the ‘prevent permute’ mode bit is set, 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 the ‘context source’ mode bit is set.

38.5.3.2.1 Dump Context

This mode may be independently specified 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.

	0 4	5	6	7	8 12	13 15	16 63
Field	—	CS	DC	PP	—	BURST SIZE	—
Reset	0	0	0	0	0	0	0
R/W	R/W						
Addr	AFEU 0x08000						

Figure 38-29. AFEU Mode Register

Table 38-29 describes AFEU mode register fields.

Table 38-29. AFEU Mode Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	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 only checked if the prevent permute bit is set. 0 Context not from FIFO 1 Context from input FIFO
6	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
7	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 or if in static assignment mode, this bit should be set to prevent AFEU from reperforming this permutation step. 0 Perform S-Box permutation 1 Do not permute
8–12	—	Reserved
13–15	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 crypto-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.
16–63	—	Reserved

38.5.3.3 AFEU Key Size Register

As displayed in Figure 38-30, this value indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in the key size register will be ignored. This register 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 is generated. If the key size register is modified during processing, a context error is generated.

	0 2	3 7	8 63
Field	—	Key Size	—
		msb<-----lsb	
Reset	0		
R/W	R/W		
Addr	AFEU 0x08008		

Figure 38-30. AFEU Key Size Register

NOTE

> The device driver will create properly formatted descriptors for situations requiring an key permute prior to ciphering. When operating the SEC as a slave (typically debug mode), the user must set the AFEU mode register to perform ‘permute with key’, write the key data to AFEU key registers, and finally write the key size to the key size register. The AFEU will start permuting the memory with the contents of the key registers immediately after the key size is written.

38.5.3.4 AFEU Context/Data Size Register

The AFEU context/data size register, shown in Figure 38-31, stores the number of bits in the final message block. This register 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.

The context/data size register 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, 0:15 = 0x1808). The write of the context size indicates to the that all context has been loaded. The user then writes the message data size to the context/data size register. After this write, the user may begin writing message data to the FIFO.

Writing to this register 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.

	0 7	8 11	12 15	16 63
Field	Data Size	—	Data Size	—
	<----lsb		msb<-----	
Reset	0			
R/W	R/W			
Addr	AFEU 0x08010			

Figure 38-31. AFEU Data Size Register

38.5.3.5 AFEU Reset Control Register

This register, as shown in Figure 38-32, allows three levels reset that affect the AFEU only, as defined by three self-clearing bits. It should be noted that the AFEU executes an internal reset sequence for hardware reset, SW_RESET, or module initialization, which performs proper initialization of the S-Box. To determine when this is complete, observe the RESET_DONE bit in the AFEU status register.

	0 4	5	6	7	8 63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	AFEU 0x08018				

Figure 38-32. AFEU Reset Control Register

Table 38-30 describes AFEU reset control register fields.

Table 38-30. AFEU Reset Control Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RI	Reset interrupt. Writing this bit active high causes AFEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the AFEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
6	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
7	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AFEU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the AFEU will enter a routine to perform proper initialization of the S-Box. 0 Do not reset 1 Full AFEU reset
8–63	—	Reserved

38.5.3.6 AFEU Status Register

This status register, shown in Figure 38-33, contains 6 bits that reflect the state of the AFEU internal signals.

The AFEU status register is read only. Writing to this location will result in address error being reflected in the AFEU interrupt status register.

	0	1	2	3	4	5	6	7	8	63
Field	—		HALT	IFW	OFR	IE	ID	RD	—	
Reset	0									
R/W	R									
Addr	AFEU 0x08028									

Figure 38-33. AFEU Status Register

Table 38-31 describes AFEU status register fields.

Table 38-31. AFEU Status Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	HALT	Halt. Indicates that the AFEU has halted due to an error. 0 AFEU not halted 1 AFEU halted **Note:** Because the error causing the AFEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
3	IFW	Input FIFO writable. The controller uses this signal to determine if the AFEU can accept the next BURST SIZE block of data. 0 AFEU Input FIFO not ready 1 AFEU Input FIFO ready **Note:** The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AFEU signals to the crypto-channel that a ‘burst size’ amount of space is available in the FIFO. The documentation of this bit in the AFEU status register is to avoid confusing a user who may read this register in debug mode.
4	OFR	Output FIFO readable. The Controller uses this signal to determine if the AFEU can source the next burst size block of data. 0 AFEU Output FIFO not ready 1 AFEU Output FIFO ready **Note:** The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AFEU signals to the crypto-channel that a 'burst size' amount of data is available in the FIFO. The documentation of this bit in the AFEU status register is to avoid confusing a user who may read this register in debug mode.
5	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, “Interrupt Status Register”). 0 AFEU is not signaling error 1 AFEU is signaling error

Table 38-31. AFEU Status Register Field Descriptions (continued)

Bits	Name	Description
6	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, “Interrupt Status Register”). 0 AFEU is not signaling done 1 AFEU is signaling done
7	RD	Reset done. This status bit, when high, indicates that AFEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8–63	—	Reserved

38.5.3.7 AFEU Interrupt Status Register

The interrupt status register, seen in Figure 38-34, tracks the state of possible errors, if those errors are not masked, through the AFEU interrupt control register.

	0	1	2	3	4	5	6	7 10	11	12	13	14	15	16 63
Field	ME	AE	OFE	IFE	—	IFO	OFU	—	IE	ERE	CE	KSE	DSE	—
Reset	0													
R/W	R													
Addr	AFEU 0x08030													

Figure 38-34. AFEU Interrupt Status Register

Table 38-32 describes AFEU interrupt status register fields.

Table 38-32. AFEU Interrupt Status Register

Bits	Names	Description
0	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
1	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
2	OFE	Output FIFO error. The AFEU output FIFO was detected non-empty upon write of AFEU data size register. 0 No error detected 1 Output FIFO non-empty error
3	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
4	—	Reserved

Table 38-32. AFEU Interrupt Status Register (continued)

Bits	Names	Description
5	IFO	Input FIFO overflow. The AFEU input FIFO has been pushed while full. 1 Input FIFO has overflowed 0 No error detected **Note:** When operating as a master, the implements flow-control, and FIFO size is not a limit to data input. When operated as a slave, the SEC cannot accept FIFO inputs larger than 512 bytes without overflowing.
6	OFU	Output FIFO underflow. The AFEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7–10	—	Reserved
11	IE	Internal error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error
12	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
13	CE	Context error. The AFEU mode register, key register, key size register, data size register, or context memory is modified while AFEU processes data. 0 No error detected 1 Context error
14	KSE	Key size error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 No error detected 1 Key size error
15	DSE	Data size error. An inconsistent value (not a multiple of 8 bits, or larger than 64 bits) was written to the AFEU data size register: 0 No error detected 1 Data size error
16–63	—	Reserved

38.5.3.8 AFEU Interrupt Control Register

The interrupt control register, shown in Figure 38-35, controls the result of detected errors. For a given error (as defined in Section 38.5.3.7, “AFEU Interrupt Status Register”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	3	4	5	6	7 10	11	12	13	14	15	16 63
Field	ME	AE	OFE	IFE	—	IFO	OFU	—	IE	ERE	CE	KSE	DSE	—
Reset	0													
R/W	R/W													
Addr	AFEU 0x08038													

Figure 38-35. AFEU Interrupt Control Register

Table 38-33 describes AFEU interrupt control register fields.

Table 38-33. AFEU Interrupt Control Register

Bits	Names	Description
0	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	AE	Address error. An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled
2	OFE	Output FIFO error. The AFEU Output FIFO was detected non-empty upon write of AFEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
3	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
4	—	Reserved
5	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
6	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
7–10	—	Reserved
11	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
12	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
13	CE	Context error. An AFEU key register, the key size register, data size register, mode register, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled

Table 38-33. AFEU Interrupt Control Register (continued)

Bits	Names	Description
14	KSE	Key size error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 Key size error enabled 1 Key size error disabled
15	DSE	Data size error. An inconsistent value was written to the AFEU data size register: 0 Data size error enabled 1 Data size error disabled
16–63	—	Reserved

38.5.3.9 AFEU End-of-Message Register

The end-of-message register in the AFEU, displayed in Figure 38-36, is used to indicate an ARC-4 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 (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 the ‘dump context’ bit in the AFEU mode register is set, the context is written to the output FIFO following the last message word. A read of this register always returns a zero value.

The AFEU end of message register is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the AFEU (through an internally generated write to the end of message register) when the SEC acts as an initiator.

	0 63
Field	AFEU End-of-Message
Reset	0
R/W	W
Addr	AFEU 0x08050

Figure 38-36. AFEU End-of-Message Register

38.5.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

38.5.3.10.1 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 the ‘prevent permutation’ mode bit is set (see Figure 38-29 on page 38-46) and the context data must be written prior to the message data. If the context registers are written during message processing or the ‘prevent permutation’ bit is not set, a context error is generated. Reading this memory while the module is not done will generate an error interrupt.

38.5.3.10.2 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 ARC-4 algorithm. If this register is written during message processing, a context error is generated.

When performing ARC-4 operations, the user has the option of performing a new S-Box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing of the next packet. The S-Box contents (256 bytes) plus the three bytes of the context memory pointers are unloaded and reloaded 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.

38.5.3.11 AFEU Key Registers

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-Box, in conjunction with the AFEU key size register. AFEU performs permutation starting with the first byte of key register 0, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations will generate an address error interrupt.

38.5.3.11.1 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 of the AFEU output FIFO. Overflows and underflows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

38.5.4 Message Digest Execution Units (MDEUs)

This section contains details about the message digest execution units (MDEUs), including a detailed register map and the modes of operation, status and control registers, and FIFOs.

The registers used in the MDEU are documented primarily for debug and slave mode operations. If the SEC requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

38.5.4.1 MDEU Mode Register

The MDEU mode register, shown in Figure 38-37, contains 8 bits that are used to program the MDEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

The mode register 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 will be generated.

	0	1 2	3	4	5	6 7	8 12	13 15	16 63
Field	Cont		INT	HMAC	PD	ALG	—	BURST SIZE	—
Reset	0								
R/W	R/W								
Addr	MDEU 0x0C000								

Figure 38-37. MDEU Mode Register

Table 38-34 describes MDEU mode register fields.

Table 38-34. MDEU Mode Register

Bits	Name	Description
0	Cont	Continue (Cont). Used during HMAC/HASH processing when the data to be hashed is spread across multiple descriptors. 0 = Do not continue—Operate the MDEU in auto completion mode. 1 = Preserve context to operate the MDEU in continuation mode.
1–2	—	Reserved
3	INT	Initialization bit (INT). Causes an algorithm-specific initialization of the digest registers. Most operations will require this bit to be set. Only static operations that are continuing from a know intermediate hash value would not initialize the registers. 0 Do not initialize 1 Initialize the selected algorithm's starting registers
4	HMAC	Identifies the hash operation to execute. 0 Perform standard hash 1 Perform HMAC operation. This requires a key and key length information.
5	PD	If set, configures the MDEU to automatically pad partial message blocks. 0 Do not autopad. 1 Perform automatic message padding whenever an incomplete message block is detected.
6–7	ALG	Message digest algorithm selection 00 = SHA-160 algorithm (full name for SHA-1) 01 = SHA-256 algorithm 10 = MD5 algorithm 11 = Reserved
8–12	—	Reserved
13–15	BURST SIZE	The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The MDEU signals to the crypto-channel that a 'burst size' amount of data is available to be pushed to the FIFO. **Note:** The inclusion of this field in the MDEU 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 MDEU.
16–63	—	Reserved

38.5.4.2 Recommended Settings for MDEU Mode Register

The most common task likely to be executed through the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPSec, and SSL/TLS. When the HMAC is being generated by a single dynamic descriptor (the MDEU acting as sole or secondary EU), the following mode register bit settings should be used:

Table 38-35. Mode Register—HMAC Generated by Single Dynamic Descriptor

Bits	Field	Value
0	Cont	0 (off)
3	Init	1 (on)
4	HMAC	1 (on)
5	PD	1 (on)

When the HMAC is being generated for a message that is spread across a chain of static descriptors, the following mode register bit settings should be used:

Table 38-36. Mode Register—HMAC Generated for a Message across a Chain of Static Descriptors

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
0	Cont	1 (on)	1 (on)	0 (off)
3	Init	1 (on)	0 (off)	0 (off)
4	HMAC	1 (on)	0 (off)	1 (on)
5	PD	0 (off)	0 (off)	1 (on)

Additional information on descriptors can be found in Section 38.4, “Data Packet Descriptors."

38.5.4.3 MDEU Key Size Register

Displayed in Figure 38-38, this value indicates the number of bytes of key memory that should be used in HMAC generation. MDEU supports at most 64 bytes of key. MDEU will generate a key size error if the value written to this register exceeds 64 bytes, or if the key length is zero and the MDEU mode register indicates an HMAC is to be performed.

	0	1 7	8 63
Field	—	Key Size	—
		msb<----lsb	
Reset	0		
R/W	R/W		
Addr	MDEU 0x0C008		

Figure 38-38. MDEU Key Size Register

38.5.4.4 MDEU Data Size Register

The MDEU data size register, shown in Figure 38-39, stores the size of the last block of data (in bits) to be processed. The first 3 bits are used to check for a bit offset in the last byte of the message. Since the engine does not support bit offsets, any value other than ‘0’ in these positions will cause a data size error. The next 3 bits 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. This register is cleared when the MDEU is reset, re-initialized, and at the end of processing the complete message.

NOTE

Writing to the data size register allows the MDEU to enter auto-start mode. Therefore, the required context data should be written prior to writing the data size.

	0 1	2 7	8 63
Field	—	Data Size	—
		msb<----lsb	
Reset	0		
R/W	R/W		
Addr	MDEU 0x0C010		

Figure 38-39. MDEU Data Size Register

38.5.4.5 MDEU Reset Control Register

This register, shown in Figure 38-40, allows three levels reset of just the MDEU, as defined by the three self-clearing bits.

	0 4	5	6	7	8 63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	MDEU 0x0C018				

Figure 38-40. MDEU Reset Control Register

Table 38-37 describes MDEU reset control register fields.

Table 38-37. MDEU Reset Control Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RI	Reset interrupt. Writing this bit active high causes MDEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the MDEU interrupt status register. 0 No reset 1 Reset interrupt logic
6	MI	Module initialization is nearly the same as software reset, except that the MDEU Interrupt control register remains unchanged. 0 No reset 1 Reset most of MDEU
7	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
8–63	—	Reserved

38.5.4.6 MDEU Status Register

This status register, as seen in Figure 38-41, contains 5 bits that reflect the state of the MDEU internal signals.

The MDEU status register is read only. Writing to this location will result in address error being reflected in the MDEU interrupt status register.

	0 1	2	3	4	5	6	7	8 63
Field	—	HALT	IFW	—	IE	ID	RD	—
Reset	0							
R/W	R							
Addr	MDEU 0x0C028							

Figure 38-41. MDEU Status Register

Table 38-31 describes MDEU status register fields.

Table 38-38. MDEU Status Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	HALT	Halt. Indicates that the MDEU has halted due to an error. 0 MDEU not halted 1 MDEU halted **Note:** Because the error causing the MDEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
3	IFW	Input FIFO writable. The Controller uses this signal to determine if the MDEU can accept the next BURST SIZE block of data. 0 MDEU Input FIFO not ready 1 MDEU Input FIFO ready **Note:** The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The MDEU signals to the crypto-channel that a 'burst size' amount of space is available in the FIFO. The documentation of this bit in the MDEU status register is to avoid confusing a user who may read this register in debug mode.
4	—	Reserved
5	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 MDEU is not signaling error 1 MDEU is signaling error
6	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, "Interrupt Status Register"). 0 MDEU is not signaling done 1 MDEU is signaling done
7	RD	Reset done. This status bit, when high, indicates that MDEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8–63	—	Reserved

38.5.4.7 MDEU Interrupt Status Register

The interrupt status register tracks the state of possible errors, if those errors are not masked, through the MDEU interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 38-42.

	0	1	2 4	5	6 10	11	12	13	14	15	16 63
Field	ME	AE	—	IFO	—	IE	ERE	CE	KSE	DSE	—
Reset	0										
R/W	R										
Addr	MDEU 0x0C030										

Figure 38-42. MDEU Interrupt Status Register

Table 38-39 describes MDEU interrupt status register fields.

Table 38-39. MDEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0	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
1	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 No error detected 1 Address error
2–4	—	Reserved
5	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 slave, the SEC cannot accept FIFO inputs larger than 512 bytes without overflowing.
6–10	—	Reserved
11	IE	Internal error. Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected **Note:** This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the error interrupt control register or by resetting the MDEU.
12	ERE	Early read error. The MDEU context was read before the MDEU completed the hashing operation. 0 No error detected 1 Early read error
13	CE	Context error. The MDEU key register, key size register, or data size register was modified while MDEU was hashing. 0 No error detected 1 Context error
14	KSE	Key size error. A value greater than 512 bits was written to the MDEU key size register. 0 No error detected 1 Key size error
15	DSE	Data size error. A value not a multiple of 512 bits while the MDEU mode register autopad bit is negated. 0 No error detected 1 Data size error
16–63	—	Reserved

38.5.4.8 MDEU Interrupt Control Register

The MDEU interrupt control register, shown in Figure 38-43, controls the result of detected errors. For a given error (as defined in Section 38.5.4.7, “MDEU Interrupt Status Register”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2 4	5	6 10	11	12	13	14	15	16 63
Field	ME	AE	—	IFO	—	IE	ERE	CE	KSE	DSE	—
Reset	0										
R/W	R/W										
Addr	MDEU 0x0C038										

Figure 38-43. MDEU Interrupt Control Register

Table 38-39 describes MDEU interrupt status register fields.

Table 38-40. MDEU Interrupt Control Register Field Descriptions

Bits	Name	Description
0	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
2–4	—	Reserved
5	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
6–10	—	Reserved
11	IE	Internal error. An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
12	ERE	Early read error. The MDEU register was read while the MDEU was performing hashing. 0 Early read error enabled 1 Early read error disabled
13	CE	Context error. The MDEU key register, the key size register, the data size register, or the mode register, was modified while the MDEU was performing hashing. 0 Context error enabled 1 Context error disabled
14	KSE	Key size error. A value outside the bounds 512 bits was written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled

Table 38-40. MDEU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
15	DSE	Data size error. An inconsistent value was written to the MDEU data size register: 0 Data size error enabled 1 Data size error disabled
16–63	—	Reserved

38.5.4.9 MDEU EU_GO Register

The EU_GO register in the MDEU, see Figure 38-44, 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 (through an internally generated write to the EU_Go register) when the SEC acts as an initiator.

	0 63
Field	MDEU EU_GO
Reset	0
R/W	W
Addr	MDEU 0x0C050

Figure 38-44. MDEU EU_GO Register

38.5.4.10 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers 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.

	0 31	32 63	
Name	**A**	**B**	Context offset 0xC100
Reset (MD5, SHA-1)	0x01234567	0x89ABCDEF	
Reset (SHA-256)	0x67E6096A	0x85AE67BB	
Name	**C**	**D**	Context Offset 0xC108
Reset (MD5, SHA-1)	0xFEDCBA98	0x76543210	
Reset (SHA-256)	0x72F36E3C	0x3AF54FA5	
Name	**E**	**F**	Context Offset 0xC110
Reset (MD5, SHA-1)	0xF0E1D2C3	0x0	
Reset (SHA-256)	0x7F520E51	0x8C68059B	
Name	**G**	**H**	Context Offset 0xC118
Reset (MD5, SHA-1)	0x0	0x0	
Reset (SHA-256)	0xABD9831F	0x19CDE05B	
Name	**Message Length Count**		Context Offset 0xC120
Reset	0		

Figure 38-45. MDEU Context Register

38.5.4.11 MDEU Key Registers

The MDEU maintains eight 64-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required. Reading any of these memory locations will generate an address error interrupt.

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.

38.5.4.12 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed. The input FIFO is multiply addressable, but those multiple addresses point only to the write (push) end of the FIFO. A write to anywhere in the MDEU FIFO address space causes the 64-bit-words to be pushed onto the MDEU input FIFO, and a read from anywhere in the MDEU FIFO address space causes the address error bit of the interrupt status register to be set.

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.

38.5.5 Random Number Generator (RNG)

This section contains details about the random number generator (RNG), including a detailed register map and the 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
- Two 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.

38.5.5.1 RNG Mode Register

The RNG mode register is used to control the RNG. One operational mode, randomizing, is defined. Writing any other value than 0 to 0:12 results in a data error interrupt that’s 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.

The mode register is cleared when the RNG is reset or re-initialized. The RNG mode register is shown in Figure 38-46.

	0 12	13 15	16 63
Field	—	Burst Count	—
Reset	0		
R/W	R/W		
Addr	0x0E000		

Figure 38-46. RNG Mode Register

Table 38-41. RNG Mode Register Definitions

Bits	Name	Description
0–12	—	Reserved, must be set to zero.
13–15	Burst count	Burst count. The 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.
16–63	—	Reserved

38.5.5.2 RNG Data Size Register

The RNG data size register is used to tell the RNG to begin generating random data. The actual contents of the data size register does not affect the operation of the RNGA. 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.

	0 63
Field	RNG Data Size
Reset	0
R/W	R/W
Addr	0x0E010

Figure 38-47. RNG Data Size Register

38.5.5.3 RNG Reset Control Register

This register, shown in Figure 38-48, contains three reset options specific to the RNG.

	0–4	5	6	7	8–63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	0x0E018				

Figure 38-48. RNG Reset Control Register

Table 38-42 describes RNG reset control register fields.

Table 38-42. RNG Reset Control Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	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
6	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
7	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the RNG. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNG reset
8–63	—	Reserved

38.5.5.4 RNG Status Register

This RNG status register, Figure 38-49, contains 4 bits that reflect the state of the RNG internal signals.

The RNG status register is read-only. Writing to this location will result in an address error being reflected in the RNG interrupt status register.

	0–1	2	3	4	5	6	7	8–63
Field	—	HALT	—	OFR	IE	—	RD	—
Reset	0							
R/W	R							
Addr	RNG 0x0E028							

Figure 38-49. RNG Status Register

Table 38-31 describes RNG status register fields.

Table 38-43. RNG Status Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	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.
3	—	Reserved
4	OFR	Output FIFO readable. The Controller uses this signal to determine if the RNG can source the next burst size block of data. 0 RNG output FIFO not ready 1 RNG output FIFO ready
5	IE	Interrupt error. This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, “Interrupt Status Register”). 0 RNG is not signaling error 1 RNG is signaling error
6	—	Reserved
7	RD	Reset done. This status bit, when high, indicates that the RNG has completed its reset sequence. 0 Reset in progress 1 Reset done
8–63	—	Reserved

38.5.5.5 RNG Interrupt Status Register

The RNG interrupt status register tracks the state of possible errors, if those errors are not masked, through the RNG interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 38-50.

	0	1	2 5	6	7 10	11	12 63
Field	ME	AE	—	OFU	—	IE	—
Reset	0						
R/W	R						
Addr	RNG 0x0E030						

Figure 38-50. RNG Interrupt Status Register

Table 38-44 describes RNG interrupt status register fields.

Table 38-44. RNG Interrupt Status Register Field Descriptions

Bits	Name	Description
0	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
1	AE	Address error. An illegal read or write address was detected within the RNG address space. 0 No error detected 1 Address error
2–5	—	Reserved
6	OFU	Output FIFO underflow. The RNG Output FIFO has been read while empty. 0 No overflow detected 1 Output FIFO has underflowed
7–10	—	Reserved
11	IE	Internal error 0 No internal error detected 1 Internal error
12–63	—	Reserved

38.5.5.6 RNG Interrupt Control Register

The RNG interrupt control register controls the result of detected errors. For a given error (as defined in Section 38.5.5.5, “RNG Interrupt Status Register”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2 5	6	7 10	11	12 63
Field	ME	AE	—	OFU	—	IE	—
Reset	0						
R/W	R/W						
Addr	RNG 0x0E038						

Figure 38-51. RNG Interrupt Control Register

Table 38-45 describes RNG interrupt status register fields.

Table 38-45. RNG Interrupt Control Register Field Descriptions

Bits	Name	Description
0	ME	Mode error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	AE	Address error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
2–5	—	Reserved
6	OFU	Output FIFO underflow. RNG Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
7–10	—	Reserved
11	IE	Internal error. An internal processing error was detected while generating random numbers. 0 Internal error enabled 1 Internal error disabled
12–63	—	Reserved

38.5.5.7 RNG EU_GO Register

The RNG EU_Go is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EUs.

	0 … 63
Field	RNG EU_GO
Reset	0
R/W	W
Addr	RNG 0x0E050

Figure 38-52. RNG EU_GO Register

38.5.5.8 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.

38.5.6 Advanced Encryption Standard Execution Units (AESUs)

This section contains details about the advanced encryption standard execution units (AESUs), including a detailed register map and the 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.

38.5.6.1 AESU Mode Register

The AESU mode register, shown in Figure 38-53, contains 3 bits that are used to program the AESU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the as an initiator. Burst size is not relevant to slave mode operations, where an external host pushes and pulls data from the execution units.

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.

	0	1	2	3	4	5 6	7	8 12	13 15	16 63
Field	ECM	—	FM	IM	RDK	CM	ED	—	BURST SIZE	—
Reset	0									
R/W	R/W									
Addr	AESU 0x12000									

Figure 38-53. AESU Mode Register

Table 38-23 describes AESU mode register fields.

Table 38-46. AESU Mode Register Field Descriptions

Bits	Name	Description
0	ECM	Extend cipher mode: Used in combination with bits 5–6 (cipher mode) to define the mode of AES operation. 0 No cipher mode extension in use, cipher mode selected by CM values 1 Extended cipher mode. Indicates AES-counter mode with CBC-MAC (AES-CCM) is in use. **Note:** Bits 5–6 (cipher mode) must be set to 00 when extend cipher mode is set, otherwise an error will be generated.
1	—	Reserved
2	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.
3	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

Table 38-46. AESU Mode Register Field Descriptions (continued)

Bits	Name	Description
4	RDK	Restore decrypt key (RDK): Specifies that key data write will contain pre-expanded key (decrypt mode only). See note on use of RDK bit. 0 Expand the user key prior to decrypting the first block 1 Do not expand the key. The expanded decryption key will be written following the context switch.
5–6	CM	Cipher mode: Controls which cipher mode the AESU will use in processing: 00 ECB -electronic codebook mode. 01 CBC- cipher block chaining mode. 10 Reserved 11 CTR- counter mode. **Note:** Bits 5–6 must be set to 00 when extend cipher mode (bit 0) is set, otherwise an error will be generated.
7	ED	Encrypt/Decrypt. If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. **Note:** This bit is ignored if CM is set to 11—CTR mode. 0 Perform decryption 1 Perform encryption
8–12	—	Reserved
13–15	BURST SIZE	The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The AESU signals to the crypto-channel that a 'burst size' amount of data is available to be pushed to the FIFO. The inclusion of this field in the AESU 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 AESU.
16–63	—	Reserved

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- AESU Key Expand Output'. The AESU mode must be 'decrypt'. See Section 38.4, “Data Packet Descriptors,” 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), 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).

38.5.6.2 AESU Key Size Register

The AESU key size register stores the number of bytes in the key (16,24,32). Any key data beyond the number of bytes in the key size register will be ignored. This register is cleared when the AESU is reset or re-initialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error will be generated. If the key size register is modified during processing, a context error will be generated.

	0 1	2 7	8 63
Field	—	KEY SIZE	—
		msb<-----lsb	
Reset	0		
R/W	R/W		
Addr	AESU 0x12008		

Figure 38-54. AESU Key Size Register

38.5.6.3 AESU Data Size Register

This AESU data size register is used to verify that the data to be processed by the AESU is divisible by the AES algorithm block size of 128 bits. The AESU does not automatically pad messages out to 128-bit blocks; therefore, any message processed by the AESU must be divisible by 128 bits or a data size error will occur.

In normal operation, the full message length to be encrypted or decrypted with the AESU is copied from the descriptor to the AESU data size register, however only bits 1–7 are checked to determine if there is a data size error. If 1–7 are all zeros, the message is evenly divisible into 128-bit blocks.

This register is cleared when the AESU is reset or re-initialized. If a data size other than 128 bits is specified, an illegal data size error will be generated. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error will be generated.

	0	1 7	8 63
Field	—	DATA SIZE	—
		msb<-----lsb	
Reset	0		
R/W	R/W		
Addr	AESU 0x12010		

Figure 38-55. AESU Data Size Register

38.5.6.4 AESU Reset Control Register

This register allows three levels reset of just AESU, as defined by the 3 self-clearing bits:

	0 4	5	6	7	8 63
Field	—	RI	MI	SR	—
Reset	0	0	0	0	0
R/W	R/W				
Addr	AESU 0x12018				

Figure 38-56. AESU Reset Control Register

Table 38-25 describes AESU reset control register fields.

Table 38-47. AESU Reset Control Register Field Descriptions

Bits	Names	Description
0–4	—	Reserved
5	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 Do not reset 1 Reset interrupt logic
6	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 Do not reset 1 Reset most of AESU
7	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register will indicate when this initialization routine is complete 0 Do not reset 1 Full AESU reset
8–63	—	Reserved

38.5.6.5 AESU Status Register

AESU status register is a read-only register that reflects the state of six status outputs. Writing to this location will result in an address error being reflected in the AESU interrupt status register.

	0	1	2	3	4	5	6	7	8	63
Field	—		HALT	IFW	OFR	IE	ID	RD	—	
Reset	0									
R/W	W									
Addr	AESU 0x12028									

Figure 38-57. AESU Status Register

Table 38-31 describes AESU status register fields.

Table 38-48. AESU Status Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	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.
3	IFW	Input FIFO writable. The Controller uses this signal to determine if the AESU can accept the next BURST SIZE block of data. 0 AESU Input FIFO not ready 1 AESU Input FIFO ready **Note:** The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AESU signals to the crypto-channel that a ‘burst size’ amount of space is available in the FIFO. The documentation of this bit in the AESU status register is to avoid confusing a user who may read this register in debug mode.
4	OFR	Output FIFO readable. The controller uses this signal to determine if the AESU can source the next burst size block of data. 0 AESU output FIFO not ready 1 AESU output FIFO ready **Note:** The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AESU signals to the crypto-channel that a 'burst size' amount of data is available in the FIFO. The documentation of this bit in the AESU status register is to avoid confusing a user who may read this register in debug mode.
5	IE	Interrupt error.This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, “Interrupt Status Register”). 0 AESU is not signaling error 1 AESU is signaling error

Table 38-48. AESU Status Register Field Descriptions (continued)

Bits	Name	Description
6	ID	Interrupt done. This status bit reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 38.7.1.4, “Interrupt Status Register”). 0 AESU is not signaling done 1 AESU is signaling done
7	RD	Reset done. This status bit, when high, indicates that AESU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8–63	—	Reserved

38.5.6.6 AESU Interrupt Status Register

The AESU interrupt status register tracks the state of possible errors, if those errors are not masked, through the AESU interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 38-58.

	0	1	2	3	4	5	6	7 10	11	12	13	14	15	16 63
Field	ME	AE	OFE	IFE	RSV	IFO	OFU	—	IE	ERE	CE	KSE	DSE	—
Reset	0													
R/W	R													
Addr	AESU 0x12030													

Figure 38-58. AESU Interrupt Status Register

Table 38-27 describes AESU interrupt register fields.

Table 38-49. AESU Interrupt Status Register Field Descriptions

Bits	Name	Description
0	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
1	AE	Address Error. An illegal read or write address was detected within the AESU address space. 0 No error detected 1 Address error
2	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
3	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

Table 38-49. AESU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
4	—	Reserved
5	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 slave, the SEC cannot accept FIFO inputs larger than 512 bytes without overflowing.
6	OFU	Output FIFO underflow. The AESU Output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7–10	—	Reserved
11	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.
12	ERE	Early read error. The AESU IV register was read while the AESU was processing. 0 No error detected 1 Early read error
13	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
14	KSE	Key size error. An inappropriate value (not 16, 24 or 32 bytes) was written to the AESU key size register 0 No error detected 1 Key size error
15	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
16–63	—	Reserved

38.5.6.7 AESU Interrupt Control Register

The AESU interrupt control register, shown in Figure 38-59, controls the result of detected errors. For a given error (as defined in Section 38.5.6.6, “AESU Interrupt Status Register”), 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.

	0	1	2	3	4	5	6	7 10	11	12	13	14	15	16 63
Field	ME	AE	OFE	IFE	RSV	IFO	OFU	—	IE	ERE	CE	KSE	DSE	—
Reset	0													
R/W	R/W													
Addr	AESU 0x12038													

Figure 38-59. AESU Interrupt Control Register

Table 38-50 describes the AESU interrupt control register fields.

Table 38-50. AESU Interrupt Control Register Field Descriptions

Bits	Name	Description
0	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
1	AE	Address error. An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
2	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
3	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
4	—	Reserved
5	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
6	IFO	Output FIFO underflow The AESU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
7–10	—	Reserved
11	IE	Internal error. An internal processing error was detected while the AESU was processing. 0 Internal error enabled 1 Internal error disabled
12	ERE	Early read error. The AESU IV register was read while the AESU was processing. 0 Early read error enabled 1 Early read error disabled

Table 38-50. AESU Interrupt Control Register Field Descriptions (continued)

Bits	Name	Description
13	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
14	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
15	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
16–63	—	Reserved

38.5.6.8 AESU End of Message Register

The AESU end of message register, shown in Figure 38-60, 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 will always return a zero value. The AESU end of message register is only used when the SEC is operated as a slave. The descriptors and crypto-channel activate the AESU (through an internally generated write to the end of message register) when the SEC acts as an initiator.

	0 63
Field	AESU End of Message
Reset	0
R/W	W
Addr	AESU 0x12050

Figure 38-60. AESU End of Message Register

38.5.6.9 AESU Context Registers

Three 64-bit context data registers 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 38-61.

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] msb<---lsb
CCM	IV[1] / MAC Tag		Encrypted MAC[2]/Decrypted MAC/Encrypted Counter		Counter [1]		Counter Modulus[1] msb<---lsb

[1] Must be written at the start of a new message.

[2] Must be written at start of new CCM decryption.

Figure 38-61. AESU Context Register

38.5.6.9.1 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 in the AESU status register as shown in Section 38.5.6.5, “AESU Status Register”. 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).

38.5.6.9.2 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 38-51

Table 38-51. Counter Modulus

Value Written	Modulus
8	2^{8}
16	2^{16}
24	2^{24}
32	2^{32}
40	2^{40}
48	2^{48}
56	2^{56}
64	2^{64}
72	2^{72}
80	2^{80}
88	2^{88}
96	2^{96}
104	2^{104}
112	2^{112}
120	2^{120}
128	2^{128}

38.5.6.9.3 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 as follows:

- Reg 1–2—Session-specific 128-bit initialization vector (from memory)
- Reg 3–4—128 bits of zero padding
- Reg 5-6—Session-specific counter (initial counter value) (from memory)
- Reg 7—Counter modulus (msb<--lsb) 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 is accomplished by the following. With the session specific key and context, the AESU performs the operations listed below.

1. Initialize the IV, and encrypt with the symmetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symmetric key.
3. Continue as in step 2 until the final block of plaintext has been processed. The result of the encryption of the final block of plaintext with the symmetric key is the MAC Tag. The full 128 bits of MAC data is written to context registers 1–2, for use in the next phase of CCM processing.

   Once the MAC tag has been generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in counter mode.
4. The first item to be encrypted in counter mode is the counter (initial counter value) from context registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC tag (retrieved from context registers 1–2) to produce the 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 802.11 frame. Note that the encrypted MAC written out to memory by the AESU is the full 128 bits. The host must only append the most-significant 64 bits to the frame as the encrypted MAC.
5. The counter value is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of plaintext to produce the first block of cipher text. The ciphertext is placed in the AESU output FIFO.
6. The counter continues to be incremented, and encrypted with the symmetric key, with the result hashed with each successive block of plaintext, until all plaintext has been converted to ciphertext. The SEC controller will manage FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the encrypted MAC has been output, the CCM encrypt operation is complete.

The context for CCM decryption/MAC generation is as follows:

- Reg 1–2—Session-specific 128-bit initialization vector (from memory)
- Reg 3–4—Encrypted MAC (from received frame) + 64 bits of zero padding
- Reg 5–6—Session-specific counter (initial counter value) (from memory)
- Reg 7—Counter modulus (msb<--lsb) 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 performs the following operations.

1. Initialize the IV, and encrypt with the symmetric key. Simultaneously, the counter (initial counter value) from context registers 5–6 is encrypted with the symmetric key. The result is hashed with the encrypted MAC (from context registers 3–4), and the resulting original MAC is written to context Reg 3-4, overwriting the encrypted MAC.

   Note that the counter is encrypted with the symmetric key, however the AESU should be set for 'decrypt' to perform the counter and CBC processes in the correct order.
2. The 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 that for both encrypt and decrypt operations, if the 802.11 frame is being processed whole (that is, not split across multiple descriptors), the initialize and final MAC bits should be set in the AESU mode register.

38.5.6.9.4 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.

38.5.6.9.5 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.

38.6 Crypto-Channels

A crypto-channel manages data associated with the one of more execution units (EUs) in the SEC. Control and data information for a given task is stored in the form of 16 32-bit word descriptors in system memory or in the crypto-channel itself. The descriptor describes how the EU should be initialized, where to fetch the data to be ciphered and where to store the ciphered data the EU outputs. Through a series of requests to the controller, the crypto-channel decodes the contents of the descriptors to perform the following functions:

- Request assignment of one or more of the several EUs in the SEC for the exclusive use of the channel.
- Automatically initialize mode registers in the assigned EU upon notification of completion of the EU reset sequence.
- Transfer data packets (up to 32 Kbytes) from system memory (master read) into assigned EU input registers and FIFOs (EU write).
- Transfer data packets (up to 32 Kbytes) from assigned EU output registers and FIFOs (EU read) to system memory space (master write).
- Automatically initialize the key size register in the assigned EU after requesting a write to EU key address space.
- Automatically initialize data size register in the assigned EU before requesting a write to EU FIFO address space.
- Automatically 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 data is complete before proceeding with further activity after writing EU_GO.
- Request assignment of the MDEU when the descriptor header calls for multi-operation processing. The MDEU will be configured to snoop input or output data intended for the primary assigned EU.
- Reset assigned EU(s).
- Release assigned EU(s).
- Automatically fetch the next descriptor from system memory and start processing, when chaining is enabled. Descriptor chains can be of unlimited size
- Provide feedback to host, through an interrupt, when a descriptor, or a chain of descriptors, has been completely processed.
- Provide feedback to host, through a modified descriptor header write back to system memory, when a descriptor, or a chain of descriptors, has been completely processed.
- Provide feedback to host, through an interrupt, when descriptor processing is halted due to an error.

- Detect static assignment of EU(s) by the controller and alter descriptor processing flow to skip EU request and EU release steps. The channel will also automatically reset the EU_DONE interrupt after receiving indication that processing of input data has been completed by the EU.

The channel waits indefinitely for the controller to complete a requested activity before continuing to process a descriptor.

38.6.1 Crypto-Channel Registers

Crypto-channel registers are described in the following sections.

38.6.1.1 Crypto-Channel Configuration Register (CCCR)

This register contains 5 operational bits permitting configuration of the crypto-channel as shown in Figure 38-62. Table 38-52 describes the CCCR.

	0 52	53 55	56 58	59	60	61	62	63
Field	—	BURST SIZE	—	WE	NE	NT	CDIE	R
Reset	0							
R/W	R/W							
Addr	Channel_1 0x02008, Channel_2 0x03008, Channel_3 0x04008, Channel_4 0x05008							

Figure 38-62. Crypto-Channel Configuration Register

Table 38-52. Crypto-Channel Configuration Register Signals

Bits	Name	Description
0–52	—	Reserved, set to zero
53–55	Burst size	The SEC implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The channel programs the various execution units to advise on space or data available in the FIFO through this field. The size of the burst is given in Table 38-53
56–58	—	Reserved, set to zero
59	WE	Writeback enable. This bit determines if the crypto-channel is allowed to notify the host of the completion of descriptor processing by setting (writing back) a DONE bit in the descriptor header. This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed. 0 Descriptor header writeback notification is disabled. 1 Descriptor header writeback notification is enabled. Header writeback notification will occur at the end of every descriptor if NOTIFICATION_TYPE is set to end-of-descriptor and Writeback_Enable is set. Writeback will occur only after the last descriptor in the chain (Next Descriptor Pointer is NIL) if NOTIFICATION_TYPE is set to end-of-chain.

Table 38-52. Crypto-Channel Configuration Register Signals (continued)

Bits	Name	Description
60	NE	Next enable. Fetch Next descriptor enable. This bit determines if the crypto-channel is allowed to request a transfer of the next descriptor, in a multi-descriptor chain, into its descriptor buffer. 0 Disable fetching of next descriptor when crypto-channel has finished processing the current one 1 Enable fetching of next descriptor when crypto-channel has finished processing the current one The address of the next descriptor in a multi-descriptor chain is either the contents of the next descriptor pointer in the descriptor buffer or the contents of the fetch register. Only if both of these registers are NIL upon completion of the descriptor currently being processed will that descriptor be considered the end of the chain.
61	NT	Notification type. Channel DONE notification type. This bit controls when the crypto-channel will generate Channel DONE Notification. 0 End-of-chain: The crypto-channel will generate channel done notification (if enabled) when it completes the processing of the last descriptor in a descriptor chain. The last descriptor is identified by having NIL loaded into both the next descriptor pointer in the descriptor buffer and the fetch register. 1 End-of-descriptor: The crypto-channel will generate channel done notification (if enabled) at the end of every data descriptor it processes Channel DONE notification can take the form of an interrupt or modified header writeback or both, depending on the state of the INTERRUPT_ENABLE and WRITEBACK_ENABLE control bits.
62	CDIE	Channel DONE interrupt enable. This bit determines whether or not the crypto-channel is allowed to assert interrupts to notify the host that the channel has completed descriptor processing. 0 Channel done interrupt disabled 1 Channel done interrupt enabled When CDIE is set, the NOTIFICATION_TYPE control bit determines when the CHANNEL_DONE interrupt is asserted. Channel error interrupts are asserted as soon as the error is detected. Refer to Section 38.6.2, "Interrupts," for complete description of crypto-channel interrupt operation.
63	R	Reset crypto-channel. This bit allows the crypto-channel to be software reset. 0 Automatically cleared by the crypto-channel when reset sequence is complete. Refer to Section 38.6.2.3, "Channel Reset," for complete description of crypto-channel reset operation. 1 Reset the registers and internal state of the crypto-channel, any EU assigned to the crypto-channel and the controller state associated with the crypto-channel.

Table 38-53 defines the burst size according to the value displayed in bits 53 through 55.

Table 38-53. Burst Size Definition

Value	Number of Dwords in Burst
000	1
001	4
010	8
011	12
100	16

Table 38-53. Burst Size Definition (continued)

Value	Number of Dwords in Burst
101	20
110	24
111	28

38.6.1.2 Crypto-Channel Pointer Status Register (CCPSR)

This register contains status fields and counters that provide the user with status information regarding the channel’s actual processing of a given descriptor.

	0–23	24–31
Field	—	STATE
Reset	0x0000_0000	
R/W	R	
Addr	Channel_1 0x02010, Channel_2 0x03010, Channel_3 0x04010, Channel_4 0x05010	

	32–36	37	38	39	40	41	42	43	44	45	46	47	48–55	56–63
Field	—	STAT	MI	MO	PR	SR	PG	SG	PRD	SRD	PD	SD	Error	PAIR_PTR
Reset	0x0000_0007													
R/W	R													
Addr	Channel_1 0x02010, Channel_2 0x03010, Channel_3 0x04010, Channel_4 0x05010													

Figure 38-63. Crypto-Channel Pointer Status Register

Table 38-54 describes the crypto-channel pointer status register fields.

Table 38-54. Crypto-Channel Pointer Status Register Signals

Bits	Name	Description
0–23	—	Reserved, set to zero
24–31	STATE	State. State of the crypto-channel state machine. This field reflects the state of the crypto-channel control state machine. The value of this field indicates exactly which stage the crypto-channel is in the sequence of fetching and processing data descriptors. Table 38-55 shows the meaning of all possible values of the STATE field. **Note:** State is documented for information only. The User will not typically care about the crypto-channel state machine.
32–36	—	Reserved, set to zero
37	STAT	Static. Crypto-channel static mode enable. 0 Crypto-channel is operating in dynamic mode. 1 Crypto-channel is operating in static mode. The STATIC bit is set when descriptor processing is initiated and the EUs indicated in the descriptor header register are already assigned to the channel. This bit is cleared when descriptor processing is initiated for the next descriptor and no EUs are assigned to the channel.

Table 38-54. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
38	Multi_EU_IN	Multi_EU_IN. The Multi_EU_IN bit reflects the type of snooping the channel will perform, as programmed by the 'snoop type' bit in the descriptor header. 0 Data input snooping by secondary EU disabled 1 Data input snooping by secondary EU enabled
39	Multi_EU_OUT	Multi_EU_OUT. The Multi_EU_OUT bit reflects the type of snooping the channel will perform, as programmed by the 'snoop type' bit in the descriptor header. 0 Data output snooping by secondary EU disabled 1 Data output snooping by secondary EU enabled
40	PR	PRI_REQ. Request primary EU assignment. 0 Primary EU assignment request is inactive. 1 The crypto-channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the Descriptor Header register as long as this bit remains set. The PRI_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_0 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the PRI_GRANT bit.
41	SR	SEC_REQ. Request secondary EU assignment. 0 Secondary EU assignment request is inactive. 1 The crypto-channel is requesting assignment of secondary EU to the channel. The channel will assert the EU request signal indicated by the Op_1 field in the descriptor header register as long as this bit remains set. The SEC_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_1 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the SEC_GRANT bit.
42	PG	Primary EU granted. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
43	SG	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.

Table 38-54. Crypto-Channel Pointer Status Register Signals (continued)

Bits	Name	Description
46	PD	Primary EU done. The PRI_DONE bit reflects the state of the done interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the done interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
48–55	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 38-56 lists the conditions that can cause a crypto-channel error and how they are represented in the ERROR field.
56–63	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 38-57 shows the meaning of all possible values of the PAIR_PTR field.

Table 38-55 shows the values of crypto-channel states.

Table 38-55. STATE Field Values

Value	Crypto-Channel State
0x00	Idle
0x01	Process_header
0x02	Fetch_descriptor
0x03	Channel_done
0x04	Channel_done_irq
0x05	Channel_done_writeback
0x06	Channel_done_notification
0x07	Channel_error
0x08	Request_pri_eu
0x09	Inc_data_pair_pointer
0x0A	Delay_data_pair_update
0x0B	Evaluate_data_pairs
0x0C	Write_reset_pri
0x0D	Release_pri_eu
0x0E	Write_reset_sec

Table 38-55. STATE Field Values (continued)

Value	Crypto-Channel State
0x0F	Release_sec_eu
0x10	Process_data_pairs
0x11	Write_mode_pri
0x12	Write_mode_sec
0x13	Write_datasize_pri
0x14	Delay_rng_done
0x15	Write_datasize_sec_multi_eu_in
0x16	Trans_request_read_multi_eu_in
0x17	Delay_pri_sec_done
0x18	Trans_request_read
0x19	Write_key_size
0x1A	Write_eu_go
0x1B	Delay_pri_done
0x1C	Write_reset_irq_pri
0x1D	Write_reset_irq_sec
0x1E	Write_datasize_sec_snoopout
0x1F	Trans_request_write_snoopout
0x20	Delay_sec_done
0x21	Trans_request_write
0x22	Evaluate_reset
0x23	Reset_write_reset_pri
0x24	Reset_release_pri_eu
0x25	Reset_write_reset_sec
0x26	Reset_release_sec_eu
0x27	Reset_channel
0x28	Write_datasize_pri_post
0x29	Reset_release_all
0x2A	Reset_release_all_delay
0x2B	Request_sec_eu
0x2C	Write_datasize_sec
0x2D	Write_pri_eu_go_multi_eu_out
0x2E	Write_sec_eu_go_multi_eu_out
0x2F	Write_pri_eu_go_multi_eu_in

Table 38-55. STATE Field Values (continued)

Value	Crypto-Channel State
0x30	Write_sec_eu_go_multi_eu_in
0x31	Write_datasize_pri_delay
0x32- 0xFF	Reserved

Table 38-56 shows the bit positions of each potential error. Multiple errors are possible.

Table 38-56. Crypto-Channel Pointer Status Register Error Field Definitions

Value	Error
b00000000	No error detected
bxxxxxxx1	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.
bxxxxxx1x	Static assignment error. Either the channel is statically assigned, but not to an EU requested by the descriptor, or the dynamic assignment request is unfillable because all suitable EUs are otherwise statically assigned.
bxxxxx1xx	Illegal descriptor header
bxxxx1xxx	Reserved
bxxx1xxxx	Pointer not complete. Caused by an invalid write to the next descriptor register in the descriptor buffer, or to the fetch register.
bxx1xxxxx	TEA. A transfer error acknowledge was received from the bus interface. When the SEC, while acting as a bus master, detects a TEA, the controller passes the TEA error to the channel in use. The channel halts and outputs an interrupt. The channel can only be restarted by resetting the channel or the whole the SEC.
bx1xxxxxx	Reserved
b1xxxxxxx	Reserved

NOTE

EU error bit (ERROR[0]) can only be cleared by first clearing the error source in the assigned EU that caused it to be set.

Table 38-57 shows the possible values of the PAIR_PTR field in the CCPSR.

Table 38-57. Crypto-Channel Pointer Status Register PAIR_PTR Field Values

Value	Error
0x01	Processing length/pointer pair 1
0x02	Processing length/pointer pair 2
0x03	Processing length/pointer pair 3
0x04	Processing length/pointer pair 4
0x05	Processing length/pointer pair 5
0x06	Processing length/pointer pair 6

Table 38-57. Crypto-Channel Pointer Status Register PAIR_PTR Field Values (continued)

Value	Error
0x07	Complete (or not yet begun) processing of header and length/pointer pairs
0x08-FF	Reserved

38.6.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The CDPR, shown in Figure 38-64, contains the address of the descriptor that the crypto-channel is currently processing. This register, along with the PAIR_PTR in the CCPSR, can be used to determine if a new descriptor can be safely inserted into a chain of descriptors.

	0 — 31	32 — 63
Field	—	CUR_DES_PTR_ADRS
Reset	0x0000_0000	
R/W	R	
Addr	Channel_1 0x02040, Channel_2 0x03040, Channel_3 0x04040, Channel_4 0x05040	

Figure 38-64. Crypto-Channel Current Descriptor Pointer Register

The bits in the current descriptor pointer register perform the functions described in Table 38-58.

Table 38-58. Crypto-Channel Current Descriptor Pointer Register Signals

Bits	Name	Description
0–31	—	Reserved, should be cleared.
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 crypto-channel requests a fetch of a descriptor from the controller. Either the value of the fetch register or of Dword 8 of the DB is transferred to the current descriptor pointer register immediately after the fetch is completed. This address will be used as destination of the write back of the modified header Dword, if header writeback notification is enabled. If a descriptor is written directly into the descriptor buffer, the host is responsible for writing a meaningful pointer value into the CURRENT_DESCRIPTOR_POINTER field.

38.6.1.4 Fetch Register (FR)

The FR, displayed in Figure 38-65, contains the address of the first byte of a descriptor to be processed. In typical operation, the host CPU creates a descriptor in memory containing all relevant mode and location information for the SEC, and 'launches' the SEC by writing the address of the descriptor to the fetch register.

Writes to the FR, while the channel is already processing a different descriptor, will be registered and held pending until the channel finishes processing the current descriptor or chain of descriptors. When the end of the current descriptor or chain of descriptors is reached, the descriptor pointed to by the FR will be treated as the next descriptor in a multi-descriptor chain. In this case, the FR must be written to before the channel begins end of descriptor notification. If the register is written after notification has begun, the descriptor will not be considered part of the current chain and will be fetched as a new stand alone descriptor or start of chain after the notification process has completed.

In summary, a channel is initiated by a direct write to the FR, and the channel always checks the FR before determining if it has truly reached the end of a chain.

	0–31	32–63
Field	—	FETCH ADRS
Reset	0x0000_0000	
R/W	R/W	
Addr	Channel_1 0x02048, Channel_2 0x03048, Channel_3 0x04048, Channel_4 0x05048	

Figure 38-65. Fetch Register

Table 38-59 describes the fetch register fields.

Table 38-59. Fetch Register Field Descriptions

Bits	Name	Description
0–31	—	Reserved, should be cleared to zero
32–63	FETCH ADRS	Fetch address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

38.6.1.5 Descriptor Buffer (DB)

The descriptor buffer (DB) actually consists of eight dword-aligned registers and the current descriptor being processed by the crypto-channel. This field is R/W enabled, however in typical operation, the DB is filled by a write from the SEC controller, acting as an initiator on the bus. (In host controlled mode, the host processor can write the entire descriptor to the DB rather than creating the descriptor in memory.)

The first dword of the DB contains the header of the descriptor under processing, and the length of the first item to be fetched. The DB uses information in the descriptor header to request and program other on-chip resources in order to complete the required security operation.

Dwords 2–7 contain fields for or one or more data pointer/length pairs. Each pair consists of a pointer register that specifies the address of the first byte of the data in system memory space, and a length register, which specifies the size if the data in bytes.

Dword 8 contains the final 'normal' length register, and an extra register referred to as the next descriptor register, which contains a pointer to the ‘next descriptor’ to be processed, if any. The pointer is set to zero for a single descriptor or the end of a multi-descriptor chain. A descriptor is considered DONE only when the contents of dword 8 have been processed by the channel. Additional information on the descriptor format and field values can be found in 38.4, “Data Packet Descriptors.”

	0	31	32	63
Dword 1	Descriptor Header	DN	Reserved	Data Length 1
Dword 2	Pointer_1		Reserved	Data Length 2
Dword 3	Pointer_2		Reserved	Data Length 3
Dword 4	Pointer_3		Reserved	Data Length 4
Dword 5	Pointer_4		Reserved	Data Length 5
Dword 6	Pointer_5		Reserved	Data Length 6
Dword 7	Pointer_6		Reserved	Data Length 7
Dword 8	Pointer_7		Next Descriptor Pointer	
Address	Channel_1 0x02080, Channel_2 0x03080, Channel_3 0x04080, Channel_4 0x05080			

Figure 38-66. Data Packet Descriptor Buffer

38.6.1.5.1 Descriptor Header

Descriptors are created by the host to guide the SEC through required crypto-graphic operations. The descriptor header defines the operations to be performed, 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 crypto-graphic operation. See Section 38.4, “Data Packet Descriptors,” for a full description of the descriptor header.

38.6.1.5.2 Descriptor Length/Pointer Pairs

The length and pointer fields represent one of seven data length/pointer pairs. Each pair defines a block of data in system memory. The length field gives the length of the block in bytes. The maximum allowable number of bytes is 32 Kbytes. A value of zero loaded into the length field indicates that this length/pointer pair should be skipped and processing continue with the next pair.

The pointer field contains the address, in global memory space, of the first byte of the data block. Transfers from the bus with the pointer address set to zero will have the length value written to the EU, and no data fetched from the bus.

38.6.1.5.3 Next Descriptor Pointer

Following the length/pointer pairs is the ‘Next Descriptor’ field, which contains the pointer to the next descriptor in memory. Upon completion of processing of the current descriptor, this value, if non-zero, is used to request a burst read of the next-data-packet descriptor. This automatic load of the next descriptor is referred to as descriptor chaining. Section 38.4, “Data Packet Descriptors,” contains a full description of the next descriptor pointer.

38.6.2 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 interrupts are always asserted to the controller. The controller can be programmed to mask channel interrupts to the host through its interrupt mask register (IMR). See Section 38.7.1.3, “Interrupt Mask Register (IMR).”

38.6.2.1 Channel Done Interrupt

The channel DONE interrupt is generated when the crypto-channel has completed processing of a single descriptor or the end of a chain of descriptors and the channel DONE interrupt enable bit in the CCCR (see Figure 38-62) is set. Which one of these conditions is responsible for the interrupt depends upon the state of the NOTIFICATION_TYPE bit in the control register, or the DONE_NOTIFICATION_FLAG in the descriptor header.

38.6.2.2 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 crypto-channel pointer status register (CCPSR). Refer to Table 38-56 for a complete listing of error types.

38.6.2.3 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.
- Setting the RESET bit in the control register while channel is statically assigned to a EU with not cause the channel to reset the assigned EU. It is the hosts responsibility to reset the assigned EU in this case.

NOTE

The CCCR and the descriptor buffer registers remain unchanged after software reset.

38.7 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 through the master/slave bus interface and to the channels and EUs through internal buses. All transfers between the host and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Arbitrate and control accesses to the bus
- Control the internal bus accesses to the EUs
- Arbitrate and assign EUs to the crypto-channels
- Monitor interrupts from channels and pass to host
- Realign initiator read data to dword boundary

38.7.1 Controller Registers

The controller registers are described in detail in the following sections.

38.7.1.1 EU Assignment Control Register (EUACR)

This register, shown in Figure 38-67, is used to make a static assignment of a EU to a particular crypto-channel. When assigned in this fashion, the EU is inaccessible to any other crypto-channel.

	0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31
Field	—	RNG	—	PKEU	—	MDEU	—	AFEU
Reset	0xF	0x0	0xF	0x0	0xF	0x0	0xF	0x0
R/W	R/W							
Addr	0x 01000							

	32 35	36 39	40 43	44 47	48 63
Field	—	DEU	—	AESU	—
Reset	0xF	0x0	0xF	0x0	0xFF00
R/W	R/W				
Addr	0x 01000				

Figure 38-67. EU Assignment Control Register (EUACR)

Table 38-60. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3

Table 38-60. Channel Assignment Value (continued)

Value	Channel
0x4	Channel 4
0xA–0xE	Undefined
0xF	Unavailable

NOTE

Writing any of the defined values shown in Table 38-60 to any of the fields in the EUACR statically assigns the EU to that specific channel. To release, the host must write 0x0 to the specified EU field of the EUACR.

38.7.1.2 EU Assignment Status Register (EUASR)

This EUASR, displayed in Figure 38-68, is used to check the assignment status (static or dynamic) of an EU to a particular crypto-channel. When an EU is already assigned, it is inaccessible to any other crypto-channel.

A 4-bit field (see Table 38-60) indicates the channel to which an EU is assigned, whether statically or dynamically.

	0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31
Field	—	RNG	—	PKEU	—	MDEU	—	AFEU
Reset	0xF	0x0	0xF	0x0	0xF	0x0	0xF	0x0
R/W	R/W							
Addr	0x 01028							

	32 35	36 39	40 43	44 47	48 63
Field	—	DEU	—	AESU	—
Reset	0xF	0x0	0xF	0x0	0xFF00
R/W	R/W				
Addr	0x 01028				

Figure 38-68. EU Assignment Status Register

38.7.1.3 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 38-69 shows the bit positions of each potential interrupt source. Each interrupt source is individually masked by setting its corresponding bit. The bit fields are described in Table 38-61.

	0	1	2	3	4	5	11	12	13	14	15
Field	CHA_2		CHA_1		A-Err	—		CHA_4		CHA_3	
Definition	Err	Dn	Err	Dn				Err	Dn	Err	Dn
Reset	0x0000										
R/W	R/W										
Addr	0x 01008										

	16	31
Field	—	
Definition		
Reset	0x0000	
R/W	R/W	
Addr	0x 01008	

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	—		PKEU		—		RNG		—		AFEU		—		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x 01008															

	48	49	50	51	52	53	54	55	56	57	58	63
Field	—		AESU		—		DEU		—	TEA	—	
Definition			Err	Dn			Err	Dn				
Reset	0x0000											
R/W	R/W											
Addr	0x 01008											

Figure 38-69. Interrupt Mask Register

38.7.1.4 Interrupt Status Register

The ISR contains fields representing all possible sources of interrupts. The interrupt status register is cleared either by a reset, or by writing the appropriate bits active in the interrupt clear register. Figure 38-70 shows the bit positions of each potential interrupt source. The bit fields are described in Table 38-61.

	0	1	2	3	4	5 11	12	13	14	15
Field	CHA_2		CHA_1		A-Err	—	CHA_4		CHA_3	
Definition	Err	Dn	Err	Dn			Err	Dn	Err	Dn
Reset	0x0000									
R/W	R									
Addr	0x01010									

	16 31
Field	—
Definition	
Reset	0x0000
R/W	R
Addr	0x01010

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	—		PKEU		—		RNG		—		AFEU		—		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x01010															

	48	49	50	51	52	53	54	55	56	57	58 63
Field	—		AESU		—		DEU		—	TEA	—
Definition			Err	Dn			Err	Dn			
Reset	0x0000										
R/W	R/W										
Addr	0x01010										

Figure 38-70. Interrupt Status Register

38.7.1.5 Interrupt Clear Register (ICR)

The interrupt control register provides a means of clearing the interrupt status register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output pin $\overline{\text{IRQ}}$ (assuming the cleared bit in the ISR is the only interrupt source). If the input source to the ISR is a steady-state signal that remains active, the appropriate ISR bit, and subsequently $\overline{\text{IRQ}}$, will be reasserted shortly thereafter. Figure 38-71 shows the bit positions of each interrupt source that can be cleared by this register. The complete bit definitions for the ICR can be found in Figure 38-71. The bit fields are described in Table 38-61.

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 that has been written with a 1.

NOTE

Interrupts are registered and sent based upon the conditions that cause them. If the cause of an interrupt is not removed, the interrupt will return a few cycles after it has been cleared using the ICR.

	0	1	2	3	4	5	11	12	13	14	15
Field	CHA_2		CHA_1		A-Err	—		CHA_4		CHA_3	
Definition	Err	Dn	Err	Dn				Err	Dn	Err	Dn
Reset	0x0000										
R/W	W										
Addr	0x01018										

	16	31
Field	—	
Definition		
Reset	0x0000	
R/W	W	
Addr	0x01018	

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	—		PKEU		—		RNG		—		AFEU		—		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x01018															

	48	49	50	51	52	53	54	55	56	57	58	63
Field	—		AESU		—		DEU		—	TEA	—	
Definition			Err	Dn			Err	Dn				
Reset	0x0000											
R/W	R/W											
Addr	0x01018											

Figure 38-71. Interrupt Clear Register

Table 38-61 describes the interrupt mask, status, and clear register fields.

Table 38-61. Interrupt Mask, Status, and Clear Register Fields

Bits	Name	Description
0–1, 2–3, 12–13, 14–15	CHA_*n*	Each of the four channels has error and done bits. The Err bit indicates whether an error has been detected: 0 No error detected 1 Error detected. The execution unit status register must be read to determine the exact cause of the error. The Dn bit indicates whether the interrupting channel or EU has completed its operation: 0 Not done 1 Done
34–35, 38–39, 42–43, 46–47, 50–51, 54–55	PKEU, RNG, AFEU, MDEU, AESU, DEU	Each of the execution units has error and done bits. The Err bit indicates whether an error has been detected: 0 No error detected 1 Error detected. The execution unit status register must be read to determine the exact cause of the error. The Dn bit indicates whether the interrupting channel or EU has completed its operation: 0 Not done 1 Done
5–11, 16–31, 36–37, 40–41, 56, 60–61	—	Reserved, set to zero.
4	A-Err	Assignment error bit. This bit indicates that a static assignment of a EU was attempted on a EU that is currently in use. 0 No error detected 1 EU assignment error detected
57	TEA	Transfer error acknowledge. Set when the SEC as a master receives a transfer error acknowledge. 0 No error detected 1 TEA detected on bus
58–63	—	Reserved, set to zero

38.7.1.6 ID Register

The read-only ID register, displayed in Figure 38-72, contains a 64-bit value that uniquely identifies the version of the SEC. The value of this register is always 0x0900_0000_0000_0000.

	0 8	9 63
Field	VERSION	
Reset	0x0900_0000_0000_0000	
R/W	R	
Addr	0x01020	

Figure 38-72. ID Register

38.7.1.7 Master Control Register (MCR)

The MCR, shown in Figure 38-73, controls certain functions in the controller and provides a means for software to reset the SEC.

	0 6	7	8 23	24 27	28	29 31
Field	—	SWR	—	Curr_Chan	GI	—
Reset	0x0000_0000					
R/W	R/W					
Addr	0x 01030					

	32 39	40 47	48 55	56 63
Field	CHA3_EU_PR_CNT	CHA4_EU_PR_CNT	CHA3_BUS_PR_CNT	CHA4_BUS_PR_CNT
Field	msb<-----lsb	msb<-----lsb	msb<-----lsb	msb<-----lsb
Reset	0x0000_0000			
R/W	R/W			
Addr	0x 01030			

Figure 38-73. Master Control Register

Table 38-62 describes the master control register signals.

Table 38-62. Master Control Register Signals

Bits	Name	Description
0–6	—	Reserved
7	SWR	Software reset. Writing 1 to this bit will cause a global software reset. Upon completion of the reset, this bit will be automatically cleared. 0 Do not reset 1 Global reset
8–23	—	Reserved
24–27	Curr_Chan	Current channel (Curr_Chan): These bits are read only. They indicate the channel number that is currently in use by the controller as a master on the bus. The possible values are: 0000 No channel is currently in use. 0001 Channel 1 is in use. 0010 Channel 2 is in use. 0011 Channel 3 is in use. 0100 Channel 4 is in use.
28	GI	Global inhibit. 0 Master will always drive GBL_B active (low). 1 Master will always drive GBL_B inactive (high).
29–31	—	Reserved

Table 38-62. Master Control Register Signals (continued)

Bits	Name	Description
32–39	CHA3_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 CHA4_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHA4_EU_PR_CTR must also be set to a different, non-zero value.
40–47	CHA4_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 CHA3_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHA3_EU_PR_CTR must also be set to a different, non-zero value.
48–55	CHA3_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 CHA4_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, CHA4_BUS_PR_CTR must also be set to a different, non-zero value.
56–63	CHA4_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 CHA3_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, CHA3_BUS_PR_CTR must also be set to a different, non-zero value.

38.7.1.8 Master Error Address Register (MEAR)

This register saves the address of the transaction whose data phase was terminated with a TEA or master parity error. A transfer error acknowledge (TEA) signal indicates a fatal error has occurred during the data phase of a bus transaction. Invalid data may have been received and stored prior to the receipt of the TEA. The channel that was initiating the transaction will be evident from that channel's error interrupt. The core may chose to reset the channel reporting the TEA, reset the whole SEC, or reset the entire system with a machine check error. In any case, the host may chose to preserve this TEA information prior to reset to assist in debug.

The MEAR only holds the address of the first error reported, in the event multiple errors are received before the first is cleared.

	0 … 31	32 … 63
Field	ADDRESS	--
Reset	0x0000_0000	
R/W	R	
Addr	0x 01038	

Figure 38-74. Master Error Address Register

Table 38-63 defines the master error address register bits.

Table 38-63. Master Error Address Register Bit Definitions

Bits	Name	Description
0–31	ADDRESS	Target address of the transaction when TEA was received.
32–63	—	Reserved

38.7.1.9 EU Access

Assignment of a EU function to a channel is done either statically or dynamically. In the case of static assignment, a EU is assigned to a channel through the EU assignment control register (EUACR). Once a EU is statically assigned to a channel, it will remain that way until the EUACR is written and the assignment is removed.

In the case of 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 a 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 issues the done signal.

38.7.1.10 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 Table 38-63.

In all cases, the controller assigns the primary EU to a requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing any grants to a channel that is requesting two EU functions.

38.7.1.11 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 CHA3_EU_PR_CNT and CHA4_EU_PR_CNT. If both CHA3_EU_PR_CNT and CHA4_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 CHA_EU_PR_CNT fields to a non-zero value will result in unpredictable operation.

38.7.1.12 Priority Arbitration

When arbitrating on the priority scheme, the priority will be as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHA3_EU_PR_CNT or CHA4_EU_PR_CNT expired
- Channel 3—Third priority, unless CHA4_EU_PR_CNT expired

- Channel 4—Lowest priority, until CHA4_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 CHA3_EU_PR_CNT and CHA4_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 CHA_EU_PR_CNT field, then that channel has the 2nd highest priority when the requested EU becomes available. CHA1 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 CHA1 operation.

It is permissible for the CHA_EU_PR_CNT values to be different from the CHA_BUS_PR_CNT values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice versa.

38.7.1.13 Round-Robin Snapshot Arbiters

The controller implements eight ‘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 through a round-robin scheme as the resource becomes available. When all requests have been satisfied, the arbiter takes another snapshot.

38.8 Bus Interface

The controller in the SEC (refer to Section 38.7, “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.

38.8.1 Bus Access

Bus access is granted through the same scheme that is used for granting EUs. When the CHA_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 38.7.1.12, “Priority Arbitration,” for more information.

38.8.1.1 Bus Master

The mechanism for transferring data to and from the bus as an initiator is either through a read request or a write request to the bus interface module. When the bus becomes available, the channel must be able to supply/take data immediately.

Slave accesses take priority over initiator accesses. It is possible that an initiator access can be interrupted (internal to the SEC) by a slave access. This occurs when a request has been made to the master interface for initiator access and a slave access occurs before the SEC is granted access to the bus. In this case, the controller saves the state of the initiator transfer, satisfies the slave access, and resumes with the initiator transfer at the point where the initiator transfer was interrupted.

38.8.1.2 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/slave interface module
5. Controller waits for bus read to begin
6. When bus read begins, controller receives data from the master/slave interface module and performs a master write to the appropriate internal address using the address supplied by the channel. Data always goes through the misalignment block.
7. Transfer continues until the bus burst read is completed and the controller has written all data to the appropriate internal address. The master/slave interface module will continue making bus requests until the full data length has been read. The master/slave interface module will also make single reads when the amount of data remaining to be fetched is less than a 60x burst (less than a full cache line.)

38.8.1.2.1 Slave Aborts

It is possible for the intended slave of an SEC master initiated transaction to terminate the transfer due to an error. Every time a Transfer Error Acknowledge is received from a slave, the TEA bit in Figure 38-70 is set, and, unless masked by Figure 38-70, the SEC channel that requested the transfer will signal an interrupt through Figure 38-70. The host will be able to determine which channel generated the interrupt by checking the ISR for the channel ERROR bit. The controller will also log the slave address that terminated with a TEA in Figure 38-70.

38.8.1.3 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. Controller acknowledges request to channel

3. Channel furnishes address, transfer length
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/slave interface module.
6. Transfer continues until the bus burst write is completed and the controller has read all data from the appropriate internal address. The master/slave interface module will continue making bus requests until the full data length has been written. The master/slave interface module will also make single reads when the amount of data remaining to be written is less than a 60x burst (less than a full cache line.) All SEC initiator writes must be dword aligned.

It is probable that several bus bursts will be required to complete any given request. When a channel has been granted access to the bus, no other internal SEC requests to the bus will be acknowledged until that transfer has been fully satisfied, that is, all bytes have been transferred.

38.8.1.3.1 Slave Access

The controller also acts as a bus slave. As a slave, the controller simply responds to read and write commands from the bus. When a write command is received from the bus, the controller takes the data from the master/slave interface module and sends it to whichever internal location is indicated by the address. For a read, the controller goes to the internal location and fetches the requested data from the specified address. Slave accesses from the bus are given priority over all other bus accesses in the SEC.

38.8.1.4 Access Size

The SEC internal memory space must be accessed on word or double word size and boundary (otherwise data may be invalid). Access size of burst will result in bus error (TEA). Accesses to the SEC internal space can be monitored through TESCR2[SEC] (refer to Section 4.3.2.9, “60x Bus Transfer Error Status and Control Register 2 (TESCR2)”).

38.8.2 Burst Size and 60x Bus Arbitration Priority

The SEC burst size is configured in the crypto-channel configuration register (CCCR[Burst Size]). Table 38-64 describes the optional burst sizes and the suggested relative priority on the 60x bus.

Table 38-64. CCCR[Burst Size] Definition and Bus Priority

CCCR[Burst size]	Number of Dwords in Burst	Bus Arbitration Priority
000	1	Any
001	4	Any
010	8	Any
011	12	Low
100	16	Low
101	20	Lowest

Table 38-64. CCCR[Burst Size] Definition and Bus Priority (continued)

CCCR[Burst size]	Number of Dwords in Burst	Bus Arbitration Priority
110	24	Lowest
111	28	Lowest

In order to optimize the performance of the SEC, it should be configured for the maximal burst size (note that this is not the same burst size as for the 60x bus, which is 32 bytes). Since the SEC holds its bus request until the transaction has ended, other masters should be assigned higher arbitration priority in order to be granted bus access during this time. Stated differently, the SEC should be configured to a low arbitration priority level (refer to Section 4.3.2.3, “60x Bus Arbitration-Level Registers (PPC_ALRH/PPC_ALRL),”) in order not to starve other bus masters. Note that the SEC bus master index is 0b0100.

38.8.3 Snooping

Cache snooping of SEC-initiated fetches can be disabled by the master control register (MCR[GI]), as described in Section 38.7.1.7, “Master Control Register (MCR)." Cache snooping is enabled by default. Refer also to Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR)."

Table 38-65. Setting Global Mode

MCR[GI]	0 All SEC transactions are global (cache snooped). 1 All SEC transactions are not global (cache snooping disabled).

The SEC internal memory space (as defined in Section 38.2, “Configuration of Internal Memory Space”) is not cacheable because this space is updated by the SEC in a non-snoopable way.

38.8.4 Interrupts

SEC interrupts are captured in SIPNR_L[SEC] and masked by SIMR_L[SEC] (masked at default). For additional information refer to Section 4.2.4, “Interrupt Vector Generation and Calculation,” Section 4.3.1.3, “CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L),” and Section 4.3.1.4, “SIU Interrupt Pending Registers (SIPNR_H and SIPNR_L).”

38.8.4.1 Interrupt Handling

All SEC channels’ interrupts are passed to SIPNR_L[SEC] (see Section 4.3.1.4). The user masks the SEC interrupt from the core by clearing and enables it by setting SIMR_L[SEC].

The relative priority of the SEC interrupt is configured by the CPM low interrupt priority register (SCPRR_L). Refer to Section 4.3.1.3, “CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L).” Setting YCCx = 111 configure the SEC to assert its request in the YCCx position. The SEC interrupt vector is 0b10_1111. Refer also to Section 4.2.4, “Interrupt Vector Generation and Calculation,” and Section 4.3.1.6, “SIU Interrupt Vector Register (SIVEC).”

The user can control which events cause an interrupt by configuring the SEC interrupt mask register. These events are:

- Done (of a channel or an execution unit)

- Error (of a channel or an execution unit)
- TEA on the 60x bus
- Execution unit assignment error

When the user detects an interrupt request from the SEC, it should further read the SEC interrupt status register (ISR) to determine the source of that interrupt. To clear an interrupt, the user should first write 1 to the bits in the SEC interrupt clear register (ICR) corresponding to the pending ISR bits, and then clear SIPNR_L[SEC] by writing 1 to it.

Events may be further masked per channel by setting or clearing the related fields in the crypto-channel configuration registers.

38.8.5 Bus Error

If a TEA event occurs, the address that caused the TEA is captured in the SEC master error address register (MEAR); refer to. If other channels are active concurrently, they continue normal operation, while the channel for which the bus error has occurred is stalled until it is reconfigured by the host processor. The transfer code (TC) of the SEC module is 0b101 (refer to Section 8.4.3.2, “Transfer Code Signals TC[0:2]”).

38.9 Power Saving Mode

The SEC may be disabled by setting SIUMCR[SECDIS] (refer to Section 4.3.2.4, “SIU Module Configuration Register (SIUMCR)”). It is enabled by default. The SEC should not be enabled/disabled during normal operation.

Appendix A
Register Quick Reference Guide

This section provides a brief guide to the core registers.

A.1 PowerPC Registers—User Registers

The implements the user-level registers defined by the PowerPC architecture except those required for supporting floating-point operations (the floating-point register file (FPRs) and the floating-point status and control register (FPSCR)). User-level, PowerPC registers are listed in Table A-1 and Table A-2 lists user-level special-purpose registers (SPRs).

Table A-1. User-Level PowerPC Registers (non-SPRs)

Description	Name	Comments	Access Level	Serialize Access
General-purpose registers	GPRs	The thirty-two 32-bit (GPRs) are used for source and destination operands.	User	—
Condition register	CR	See the *Programming Environments Manual*	User	Only **mtcrf**

Table A-2 lists SPRs defined by the PowerPC architecture implemented on the MPC8272.

Table A-2. User-Level PowerPC SPRs

SPR Number			Name	Comments	Serialize Access
Decimal	SPR [5–9]	SPR [0–4]			
1	00000	00001	XER	See the *Programming Environments Manual*	Write: Full sync Read: Sync relative to load/store operations
8	00000	01000	LR	See the *Programming Environments Manual*	No
9	00000	01001	CTR	See the *Programming Environments Manual*	No
268	01000	01100	TBL read [1]	See the *Programming Environments Manual*	Write (as a store)
269	01000	01101	TBU read [2]		

[1] Extended opcode for mftb, 371 rather than 339.

[2] Any write (**mtspr**) to this address causes an implementation-dependent software emulation exception.

A.2 PowerPC Registers—Supervisor Registers

All supervisor-level registers implemented on the MPC8272 are SPRs, except for the machine state register (MSR), described in Table A-3.

Table A-3. Supervisor-Level PowerPC Registers

Description	Name	Comments	Serialize Access
Machine state register	MSR	See the *Programming Environments Manual* and *MPC603e RISC Microprocessor User's Manual*	Write fetch sync

Table A-4 lists supervisor-level SPRs defined by the PowerPC architecture.

Table A-4. Supervisor-Level PowerPC SPRs

SPR Number			Name	Comments	Serialize Access
Decimal	SPR[5–9]	SPR[0–4]			
18	00000	10010	DSISR	See the *Programming Environments Manual*	Write: Full sync Read: Sync relative to load/store operations
19	00000	10011	DAR	See the *Programming Environments Manual*	Write: Full sync Read: Sync relative to load/store operations
22	00000	10110	DEC	See the *Programming Environments Manual*	Write
26	00000	11010	SRR0	See the *Programming Environments Manual*	Write
27	00000	11011	SRR1	See the *Programming Environments Manual*	Write
272	01000	10000	SPRG0	See the *Programming Environments Manual*	Write
273	01000	10001	SPRG1		
274	01000	10010	SPRG2		
275	01000	10011	SPRG3		
284	01000	11100	TBL write[1]	See the *Programming Environments Manual*	Write (as a store)
285	01000	11101	TBU write[1]		
287	01000	11111	PVR	Section 2.3.1.2.4, "Processor Version Register (PVR)."	No (read-only register)

[1] Any read (**mftb**) to this address causes an implementation-dependent software emulation exception.

A.3 MPC8272-Specific SPRs

Table A-2 and Table A-5 list SPRs specific to the MPC8272. Supervisor-level registers are described in Table A-5.

Table A-5. . MPC8272-Specific Supervisor-Level SPRs

SPR Number			Name	Comments	Serialize Access
Decimal	SPR[5–9]	SPR[0–4]			
976	11110	10000	DMISS	See the *MPC603e RISC Microprocessor User's Manual*	
977	11110	10001	DCMP	See the *MPC603e RISC Microprocessor User's Manual*	
978	11110	10010	HASH1	See the *MPC603e RISC Microprocessor User's Manual*	
979	11110	10011	HASH2	See the *MPC603e RISC Microprocessor User's Manual*	
980	11110	10100	IMISS	See the *MPC603e RISC Microprocessor User's Manual*	
981	11110	10101	ICMP	See the *MPC603e RISC Microprocessor User's Manual*	
982	11110	10110	RPA	See the *MPC603e RISC Microprocessor User's Manual*	
1008	11111	10000	HID0	Section 2.3.1.2.1, "Hardware Implementation-Dependent Register 0 (HID0)."	
1009	11111	10001	HID1	Section 2.3.1.2.2, "Hardware Implementation-Dependent Register 1 (HID1)."	
1010	11111	10010	IABR	See the *MPC603e RISC Microprocessor User's Manual*	
1011	11111	10011	HID2	Section 2.3.1.2.3, "Hardware Implementation-Dependent Register 2 (HID2)."	

Appendix B
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8272 PowerQUICC II Family Reference Manual*. This is the initial printed version of the manual so there are currently no differences.

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this book. Some of the terms and definitions included in the glossary are reprinted from *IEEE Std. 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*, copyright ©1985 by the Institute of Electrical and Electronics Engineers, Inc. with the permission of the IEEE.

Note that some terms are defined in the context of how they are used in this book.

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*.

Asynchronous exception. *Exceptions* that are caused by events external to the processor's execution. In this document, the term 'asynchronous exception' is used interchangeably with the word *interrupt*.

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

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.

Blockage. A pipeline stall that occurs when an instruction occupies an execution unit and prevents a subsequent instruction from being dispatched.

Boundedly undefined. A characteristic of results of certain operations 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 bus transfer whose data phase consists of a sequence of transfers. For example, on a 64-bit bus, a four-beat burst can transfer four, 64-bit double words.

Bus parking. A feature that optimizes the use of the bus by allowing a device to retain bus mastership without having to rearbitrate.

C

Cache . High-speed memory component containing recently-accessed data and/or instructions (subset of main memory).

Cache coherency. An attribute in which an accurate and common view of memory is provided to all devices that share the a 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-outs. *Cache blocks* 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.

Clear. To cause a bit or bit field to register a value of zero. The opposite of *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. An operation in which modified data in a *cache block* is copied back to memory.

Critical-data first. An aspect of *burst* accesses that allow the requested data (typically a word or double word) in a *cache block* to be transferred first.

D

Denormalized number. A nonzero floating-point number whose *exponent* has a reserved value, usually the format's minimum, and whose explicit or implicit leading significand bit is zero.

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.

Direct-store. Interface available on processors that implement the PowerPC architecture only to support direct-store devices from the POWER architecture. When the T bit of a *segment descriptor* is set, the descriptor defines the region of memory that is to be used as a direct-store segment. Note that this facility is being phased out of the architecture and will not likely be supported in future devices. Therefore, software should not depend on it and new software should not use it.

E

Effective address (EA). The 32- or 64-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.

Exception. A condition encountered by the processor that requires special, supervisor-level processing.

Exception handler. A software routine that executes when an exception is taken. Normally, the exception handler corrects the condition that caused the exception, or performs some other meaningful task (that may include aborting the program that caused the exception). The address for each exception handler is identified by an exception vector offset defined by the architecture and a prefix selected via the MSR.

Extended opcode. A secondary opcode field generally located in instruction bits 21–30, that further defines the instruction type. All PowerPC instructions are one word in length. The most significant 6 bits of the instruction are the *primary opcode*, identifying the type of instruction. *See also* Primary opcode.

Execution synchronization. A mechanism by which all instructions in execution are architecturally complete before beginning execution (appearing to begin execution) of the next instruction. Similar to context synchronization but doesn't force the contents of the instruction buffers to be deleted and refetched.

Exponent. In the binary representation of a floating-point number, the exponent is the component that normally signifies the integer power to which the value two is raised in determining the value of the represented number. *See also* Biased exponent.

F

Fetch. Retrieving instructions from either the cache or main memory and placing them into the instruction queue.

Fully-associative. Addressing scheme where every cache location (every byte) can have any possible address.

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.

H

Harvard architecture. An architectural model featuring separate caches for instruction 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 arithmetic.

Illegal instructions. A class of instructions that are not implemented for a particular processor that implement the PowerPC architecture. 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.

Implementation-dependent. An aspect of a feature in a processor's design that is defined by a processor's design specifications rather than by the PowerPC architecture.

Implementation-specific. An aspect of a feature in a processor's design that is not required by the PowerPC architecture, but for which the PowerPC architecture may provide concessions to ensure that processors that implement the feature do so consistently.

Imprecise exception. A type of *synchronous exception* that is allowed not to adhere to the precise exception model (*see* Precise exception). The PowerPC architecture allows only floating-point exceptions to be handled imprecisely.

Internal bus. The bus connecting the core and system interface unit (SIU).

Instruction latency. The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

Interrupt. An *asynchronous exception*. On PowerPC processors, interrupts are a special case of exceptions. *See also* asynchronous exception.

L

Latency. The time an operation requires. For example, execution latency is the number of processor clocks an instruction takes to execute. Memory latency is the number of bus clocks needed to perform a memory operation.

Least-significant bit (lsb). The bit of least value in an address, register, 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.

M

Master, The name given to a bus device that has been granted control, or mastership, of the bus.

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 controller. A unit whose primary function is to control the external bus memories and I/O devices.

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.

Microarchitecture. The hardware details of a microprocessor’s design. Such details are not defined by the PowerPC architecture.

Mnemonic. The abbreviated name of an instruction used for coding.

Modified state. When a cache block is in the modified state, it has been modified by the processor since it was copied from memory. *See* MESI.

Munging. A modification performed on an *effective address* that allows it to appear to the processor that individual aligned scalars are stored as *little-endian* values, when in fact it is stored in *big-endian* order, but at different byte addresses within double words. Note that munging affects only the effective address and not the byte order. Note also that this term is not used by the PowerPC architecture.

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

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

O

OEA (operating environment architecture). The level of the architecture that describes PowerPC memory management model, supervisor-level registers, synchronization requirements, and the exception model. It also defines the time-base feature from a supervisor-level perspective. Implementations that conform to the PowerPC OEA also conform to the PowerPC UISA and VEA.

Optional. A feature, such as an instruction, a register, or an exception, that is defined by the PowerPC architecture but not required to be implemented.

Out-of-order. An aspect of an operation that allows it to be performed ahead of one that may have preceded it in the sequential model, for example, speculative operations. An operation is said to be performed out-of-order if, at the time that it is performed, it is not known to be required by the sequential execution model. *See* In-order.

Out-of-order execution. A technique that allows instructions to be issued and completed in an order that differs from their sequence in the instruction stream.

Overflow. An error condition that occurs during arithmetic operations when the result cannot be stored accurately in the destination register(s). For example, if two 32-bit numbers are multiplied, the result may not be representable in 32 bits.

P

Pace control. Controls the rate of the data flow between a master and slave.

Page. A region in memory. The OEA defines a page as a 4-Kbyte area of memory, aligned on a 4-Kbyte boundary.

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 processors that implement the PowerPC architecture, a page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.

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. S*ee* Secondary opcode.

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.

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.

Real address mode. An MMU mode when no address translation is performed and the *effective address* specified is the same as the physical address. The processor’s MMU is operating in real address mode if its ability to perform address translation has been disabled through the MSR registers IR and/or DR bits.

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.

Register indirect addressing. A form of addressing that specifies one GPR that contains the address for the load or store.

Register indirect with immediate index addressing. A form of addressing that specifies an immediate value to be added to the contents of a specified GPR to form the target address for the load or store.

Register indirect with index addressing. A form of addressing that specifies that the contents of two GPRs be added together to yield the target address for the load or store.

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.

Reserved field. In a register, a reserved field is one that is not assigned a function. A reserved field may be a single bit. The handling of reserved bits is *implementation-dependent*. Software is permitted to write any value to such a bit. A subsequent reading of the bit returns 0 if the value last written to the bit was 0 and returns an undefined value (0 or 1) otherwise.

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

Scalability. The capability of an architecture to generate *implementations* specific for a wide range of purposes, and in particular implementations of significantly greater performance and/or functionality than at present, while maintaining compatibility with current implementations.

Scan chain. The peripheral buffers of a device, linked in JTAG test mode, that are addressed in a shift-register fashion.

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 any 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.

Significand. The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.

Slave. A device that responds to the master’s address. A slave receives data on a write cycle and gives data to the master on a read cycle.

Static branch prediction. Mechanism by which software (for example, compilers) can give a hint to the machine hardware about the direction a branch is likely to take.

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.

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.

U

UISA (user instruction set architecture). The level of the architecture to which user-level software should conform. The UISA defines the base user-level instruction set, user-level registers, data types, floating-point memory conventions and exception model as seen by user programs, and the memory and programming models.

User mode. The unprivileged operating state of a processor used typically by application software. In user mode, software can only access certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

V

VEA (virtual environment architecture). The level of the *architecture* that describes the memory model for an environment in which multiple devices can access memory, defines aspects of the cache model, defines cache control instructions, and defines the time-base facility from a user-level perspective. *Implementations* that conform to the PowerPC VEA also adhere to the UISA, but may not necessarily adhere to the OEA.

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

Watchpoint. An event that is reported, but does not change the timing of the machine.

Word. A 32-bit data element. Note that on other processors a word may be a different size.

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

Part V—Integrated Security Engine V